Computing and Informatics, Vol. 38, 2019, 1009–1038, doi: 10.31577/cai_2019_5_1009

A MORE FAITHFUL FORMAL DEFINITION OF THE DESIRED PROPERTY FOR DISTRIBUTED SNAPSHOT ALGORITHMS TO MODEL CHECK THE PROPERTY

Ha Thi Thu Doan, Kazuhiro Ogata

School of Information Science
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi,
Ishikawa, 923-1292, Japan
e-mail: {doanha, ogata}@jaist.ac.jp

Abstract. The first distributed snapshot algorithm was invented by Chandy and Lamport: Chandy-Lamport distributed snapshot algorithm (CLDSA). Distributed snapshot algorithms are crucial components to make distributed systems fault tolerant. Such algorithms are extremely important because many modern key software systems are in the form of distributed systems and should be fault tolerant. There are at least two desired properties such algorithms should satisfy: 1) the distributed snapshot reachability property (called the DSR property) and 2) the ability to run concurrently with, but not alter, an underlying distributed system (UDS). This paper identifies subtle errors in a paper on formalization of the DSR property and shows how to correct them. We give a more faithful formal definition of the DSR property; the definition involves two state machines – one state machine M_{UDS} that formalizes a UDS and the other M_{CLDSA} that formalizes the UDS on which CLDSA is superimposed (UDS-CLDSA) – and can be used to more precise model checking of the DSR property for CLDSA. We also prove a theorem on equivalence of our new definition and an existing one that only involves M_{CLDSA} to guarantee the validity of the existing model checking approach. Moreover, we prove the second property, namely that CLDSA does not alter the behaviors of UDS.

Keywords: Distributed snapshot algorithm, reachability, state machine, property specification, model checking

Mathematics Subject Classification 2010: 68N30

1 INTRODUCTION

Many modern key software systems are in the form of distributed systems [1, 2], which consist of many components coordinating their actions by passing messages and working together to achieve a common goal. The modern distributed applications, such as cloud computing [3] and web search [4], are getting more complicated. Such systems should be fault tolerant because they need to run for a long time, keeping on providing services to users, other systems, etc. To make distributed systems fault tolerant, it is necessary to use many non-trivial distributed algorithms, such as snapshot algorithms and self-stabilizing algorithms. Distributed snapshot algorithms (DSAs) deal with a significant problem, recording global states of a distributed system, which helps solving others, such as recovering from faulty states and detecting stable properties. Therefore, DSAs become the core of many fault tolerant distributed systems. However, the challenge is how to determine a consistent global state. To clearly record consistent global states, a DSA should satisfy two properties:

1. the distributed snapshot reachability property (called the DSR property) and
2. the ability to run concurrently with, but not alter, an underlying distributed system (UDS).

Considered as the most important desired property of the algorithm, the DSR property is as follows: Let s_1 be the state in which a DSA initiates, s_2 be the state in which DSA terminates, and s_* be the snapshot taken, then s_* is reachable from s_1 and s_2 is reachable from s_*.

Because distributed systems are usually complicated and it is hard to ensure their reliability, the formal verification of distributed systems is essential. Model checking [5, 6] is a popular automatic formal technique for verifying state transition systems. Many model checkers, such as Spin [7], NuSMV[8], and TLA+ [9], have been developed in order to formally verify various kinds of software and hardware systems. Model checking is highly suitable for formally verifying distributed systems. Many researchers [10, 11, 12] are interested in model checking of distributed systems. However, the application of model checking to verification of DSAs has not been fully investigated due to the specific characteristic of DSAs (they are superimposed on, but do not interfere with UDSs).

This paper focuses on Chandy-Lamport distributed snapshot algorithm (CLDSA) [13] that is the first such algorithm. Several variants of CLDSA, such as Spezialetti-Kearns algorithm [14] and Venkatesans incremental snapshot algorithm [15], have been proposed. In an attempt to formally analyze CLDSA, the authors in [16] have used Maude [17] to model check that CLDSA enjoys the DSR property. Maude is a language and a system supporting executable specification and declarative programming in rewriting logic. Two model checking facilities equipped with Maude are the LTL model checker and the search command. In [16], the Maude search command is used. However, the DSR property is encoded in terms of the Maude search command and the encoding does not reflect the informal description of the property originally given in [13]. We recognize that the informal description of the DSR property involves both a UDS and the UDS on which CLDSA is superimposed (UDS-CLDSA), while their definition of the property involves only the UDS-CLDSA. Consequently, we do not think that the existing study provides a good foundation for meaningful model checking of the DSR property for CLDSA. Therefore, it is necessary to express the DSR property accurately and then consider the equivalence between the new formal definition and the existing one.

The challenge is that the DSR property relates to two different systems: a UDS and the UDS-CLDSA. It is not straightforward to express the property in typical existing temporal logics, such as linear temporal logic (LTL) and computation tree logic (CTL) because such a temporal logic only takes into account one state machine or a Kripke structure. The authors of the technical paper [18] attempt to find a way to express the DSR property more faithfully. However, the technical report [18] has not been reviewed. We detected flaws lurking in their formalization.

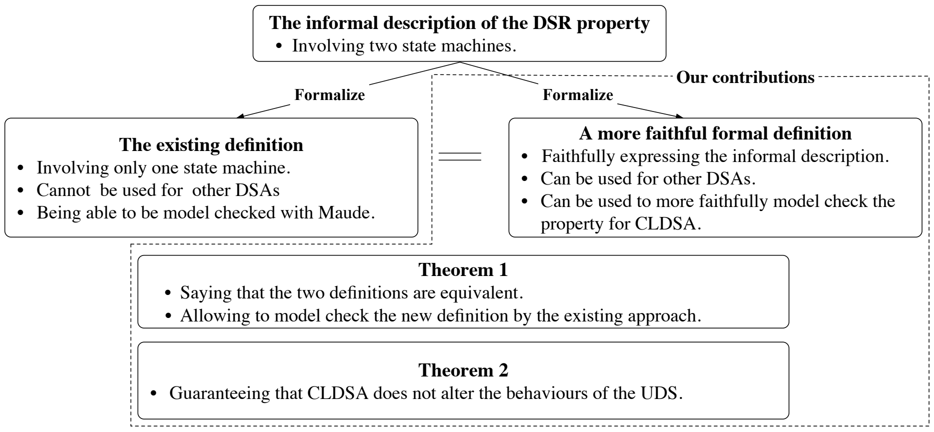

The informal description of the DSR property
• Involving two state machines.
Our contributions
Formalize
Formalize
The existing definition
• Involving only one state machine.
• Cannot be used for other DSAs
• Being able to be model checked with Maude.
A more faithful formal definition
• Faithfully expressing the informal description.
• Can be used for other DSAs.
• Can be used to more faithfully model check the property for CLDSA.
Theorem 1
• Saying that the two definitions are equivalent.
• Allowing to model check the new definition by the existing approach.
Theorem 2
• Guaranteeing that CLDSA does not alter the behaviours of the UDS.

Figure 1. The contributions of the paper

This paper is an extended and revised version of the paper [19], which shows a thorough study on formal expression of the DSR property. We have pointed out mistakes in [18] in detail. Correcting these mistakes, we formalize a UDS and the UDS-CLDSA as state machines more precisely. Carefully investigating the informal description of the DSR property, we give a formal definition of the property, which is more likely to faithfully express the informal description and can be used to more precisely model check of the CLDSA property. The essential of our method to formalize a UDS and the UDS-CLDSA is that the formalization of a UDS-CLDSA is generated from the one of the UDS. This is because we take into account an important aspect of the algorithm, namely that the algorithm runs concurrently with, but does not alter, the behaviors of UDS. Not only snapshot algorithms but also many other distributed algorithms (called control algorithms [21]), such as checkpointing algorithms and termination detection algorithms, do not alter the behaviors of the UDS. These algorithm executions are superimposed on the underlying application execution, but do not interfere with the application execution. Therefore, one considerable contribution of our research is a method by which the class of these algorithms can be precisely formalized, and then any of their desired properties that are related to both a UDS and the UDS on which these algorithms are superimposed could be formally expressed. Furthermore, we prove that our new definition is equivalent to the existing one in [16] to confirm the validity of the model checking approach. It is not straightforward to directly model check the new definition with an existing model checker because two different state machines should be taken into account, while existing model checkers, such as Spin, NuSMV, and TLC, support model checking for systems that can be formalized as only one state machine. The equivalence of the two definitions suggests we can use an existing model checker to tackle the DSR property. Moreover, we prove that CLDSA runs concurrently, but does not alter, a UDS as the second property. More on our contributions are depicted in Figure 1.

The rest of the paper is organized as follows. The next section introduces some preliminaries. Section 3 describes how to formalize a UDS and the UDS-CLDSA as state machines. Section 4 gives the new definition of the DSR property. Section 5 presents the theorem that guarantees the validity of the existing model checking approach. Section 6 proves the theorem on simulation between M_{CLDSA} and M_{UDS} to confirm the second property. Section 7 mentions related work, and Section 8 concludes the paper.

2 PRELIMINARIES

2.1 Chandy-Lamport Distributed Snapshot Algorithm (CLDSA)

CLDSA works by using control messages called *markers*. Each process can record its local state when it has not yet received any marker from its incoming channels. It then sends one marker along each of its outgoing channels. Because the channel is FIFO, the marker acts as a separator for the messages in the channel to determinate the sequence of messages that should be recorded in the state of the channel. There are two rules by which a process will execute the marker-sending rule on recording its local state and the marker-receiving rule on receiving a marker, respectively. The outline of the algorithm is as follows:

Marker-Sending Rule for a process p: for each its outgoing channel c, p sends one marker along c after recording its state and before sending further messages along c.

Marker-Receiving Rule for a process p: when the process p gets a marker from one of its incoming channels c,

- **if** p has not yet recorded its state **then** p records its state according to Marker-Sending Rule for p and the state of channel c as an empty sequence
- **else** p records the state of c as the sequence of messages received along c after recording p's state and before receiving the marker along c.

The algorithm will terminate when each process has already recorded its state and the states of all of its incoming channels. The global snapshot is made by combining those recorded process and channel states.

2.2 State Machine

A state machine consists of a set of states, some of which are initial states, and a binary relation over states. The definition is as follows:

Definition 1 (State Machine)**.** A state machine $M \triangleq \langle S, I, T \rangle$ consists of

1. a set of states S;
2. a set of initial states $I \subseteq S$;
3. a total binary relation[1] over states $T \subseteq S \times S$.

Elements (s, s') of the binary relation are called state transitions, where (s, s') says that s can change to s'. Since T is total, for each state $s \in$ S there exists a state $s' \in S$ such that $(s, s') \in T$. The reachable states of a state machine are inductively defined as follows:

1. each initial state is reachable and
2. for each reachable state s and each state transition (s, s'), s' is reachable.

A state predicate p is an invariant property of the state machine if and only if $p(s)$ holds for all reachable states s of the state machine.

2.3 Maude

Maude [17] is a specification and programming language based on rewriting logic. The basic units of specifications and programs are modules. A module contains syntax declarations, providing a suitable language to describe a system. A module

[1] Standard definitions of state machines do not require that T is total, but it is convenient to be able to generate infinite state sequences from state machines for proving that one state machine can simulate another one and then we assume that T is total. Note that a Kripke structure, which is used to define semantics of temporal logics and includes a state machine, requires that T is total and then our assumption would be reasonable.

consists of sort, sub-sort, operator, variable, equation and membership declarations, as well as rewrite rule declarations. Note that membership declarations are not used at all in this paper.

A sort denotes a set, corresponding to a type in conventional programming languages. For example, the sort `Nat` denotes the set of natural numbers. There is the relation among sorts which is the same as the subset relation. A sort is a subsort of another sort if and only if the set denoted by the former is a subset of the one by the latter, and the latter is called a supersort of the former. Let `Zero` and `NzNat` be the sorts denoting $\{0\}$ and the set of non-zero natural numbers, and then both of which are subsets of the set of natural numbers. `Zero` and `NzNat` are subsorts of `Nat` and `Nat` is a supersort of `Zero` and `NzNat`. An operator is declared as follows: $f : S_1 \ldots S_n \to S$, where $S_1, \ldots, S_n, S$ are sorts and $n \geq 0$. Note that `->` may be used instead of $\to$. $S_1 \ldots S_n$, S and $S_1 \ldots S_n \to S$ are called the arity, the sort and the rank of f, respectively. When $n = 0$, f is called a constant. f takes a sequence of n things of $S_1, \ldots, S_n$ and makes something of S, where "things" and "something" are what are called terms and will be described soon. Operators denote functions or data constructors. Each variable has its own sort. Terms of a sort S are inductively defined as follows:

1. each variable whose sort is S is a term of S and
2. for each operator f whose rank is $S_1 \ldots S_n \to S$ and n terms $t_1, \ldots, t_n$ of $S_1 \ldots S_n$, $f(t_1, \ldots, t_n)$ is a term of S.

Note that when $n = 0$, f as it is a term of S. An operator may contain underscores, such as `_+_ : Nat Nat -> Nat`. If that is the case, a different notation than $f(t_1, \ldots, t_n)$ is used. For example, if `X` is a variable of `Nat`, then `X + X` is a term of `Nat`. Use of operators containing underscores allows us to define context-free grammars. An equation declares that two terms are equal. A rewrite rule declares that a term changes to another one. Equations can be used to define functions, while rewrite rules can be used to define state transitions. A sort and the set denoted by the sort are interchangeably used in this paper.

3 THE FORMALIZATIONS OF A UDS AND THE UDS-CLDSA

3.1 Formalizing a UDS as a State Machine

A UDS consists of a finite set of processes that are connected and communicated by channels. This paper considers snapshot algorithms suited for FIFO channels. Channels are assumed to deliver messages in the order sent. Messages are delivered reliably without any error in finite but arbitrary time. There may be more than one channel from one process to another. A UDS can be described as a labeled, directed graph in which the vertices represent the processes and the directed edges represent the channels. Figure 2 describes a UDS consisting of three processes p, q, r and four channels $c1$, $c2$, $c3$, $c4$. The channel $c4$ is used to directly send messages from r to p, but not vice versa.

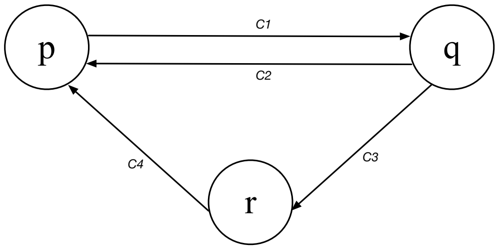


Figure 2. A distributed system with processes p, q, r and channels $c1$, $c2$, $c3$, $c4$

A global state of a distributed system is a collection of component process and channel states. The state of a channel is characterized by the sequence of in-trans messages sent along the channel and have not yet received by the destination process. The initial global state is one in which each process is in its initial state and the state of each channel is the empty sequence.

Since state machines are suitable to formalize concurrent systems, they are used to formalize a UDS and the UDS-CLDSA in our research. In this paper, Maude notation is used to describe state machines. Let M_{UDS} be the state machine that formalizes a UDS. We first consider how to express the states of a UDS and the state transitions of M_{UDS}.

Because our aim is to model check desired properties for CLDSA, we suppose that the reachable states of a UDS are bounded and so are the reachable states of the UDS-CLDSA. Note that the whole states of a UDS may be unbounded and so may be the whole states of the UDS-CLDSA. Because we formalize a channel from one process to another as an inductively defined queue, the whole states of a UDS are unbounded. However, the reachable states of the UDS could be bounded, for example, by preventing new messages from being generated.

3.1.1 State Expression

The states of process and channel are described by *name: value* pairs (called observable components), where *name* may have parameters: *p-state[_]:_* for process, where the parameter of *name* is a process identifier and *value* is the state of the process and *c-state[_,_,_]:_* for channel, where the three parameters are a source process, a destination process and a natural number, respectively, and *value* is the state of the channel. OCom is the sort for those observable components. These observable components are expressed by the following operators that are constructors as specified with *ctor*:

op p-state[_]:_ : Pid PState → OCom [ctor].

op c-state[_, _, _]:_ : Pid Pid Nat MsgQueue$P$ → OCom [ctor].

where Pid, PState, Nat and MsgQueue are the sorts for process identifiers, process states, natural numbers and queues of messages for which the sort Msg is used, respectively.

Let p, $q \in$ Pid, $ps \in$ PState, $n \in$ Nat and $ms \in$ MsgQueue. (p-state$[p]$: ps) is an observable component whose name is p-state$[p]$ where p is a parameter and whose value is ps, expressed as a term of OCom that says that the local state of a process p is ps. (c-state$[p, q, n]$: ms) is an observable component whose name is c-state$[p, q, n]$ where p, q, n are parameters and whose value is ms, expressed as a term of OCom that says that the state of a channel from the process p to the process q is ms. Since there may be more than one channel from p to q, a natural number n is used in (c-state$[p, q, n]$: ms) to identify the channel. We use an associative-commutative collection (called a soup) to express global states of a UDS. The corresponding sort is ConFigure. The following operators are prepared to construct the sort.

subsort OCom < Config,

op empConfig : $\to$ Config [ctor],

op _ _ : Config Config $\to$ Config [ctor assoc comm id:empConfig]

where empConfig denotes the empty soup of observable components. Config is a super-sort of OCom, which means that each term of OCom is treated as the singleton soup only consisting of the term. The juxtaposition operator _ _ is used to construct soups of observable components. For $c_1, c_2 \in$ Config, $c_1 c_2 \in$ ConFigure. The juxtaposition operator is associative and commutative as specified with *assoc* and *comm*, and empConfig is an identity of the operator specified with id:empConfig.

3.1.2 State Transitions

Each process in a UDS may do three kinds of actions:

i. Change of Process State: it may change its state without sending or receiving any message,
ii. Sending of Message: it may send a message by putting the message into one of its outgoing channels and may change its state (or may not change its state), and
iii. Receipt of Message: it may get the top message from one of its incoming channels if the channel is not empty and may change its state (or may not change its state).

These actions are described as transition rules. A transition rule is described in the form of rewrite rules[2]. In the following part, $P, Q \in$ Pid, $PS1, PS2 \in$ PState,

[2] Each rewrite rule (and each equation) should be executable and the rewrite rule should be split into multiple executable ones so that model checking can be doable with Maude. Since the main purpose of the paper is to give a more faithful definition of the DSR property and confirm the validity of the model checking approach used in the existing study, we make each rewrite rule as general as possible to cover all possible situations.

$N \in$ Nat, $CS \in$ MsgQueue and $M \in$ Msg are variables of those sorts.

- Change of Process State is described as the following transition rule:

  $$(\text{p-state}[P] : PS1) \Rightarrow (\text{p-state}[P] : PS2).$$

- Sending of Message is described as the following transition rule:

  $$(\text{p-state}[P] : PS1)(\text{c-state}[P, Q, N] : \text{CS}) \Rightarrow$$
  $$(\text{p-state}[P] : PS2)(\text{c-state}[P, Q, N] : \text{enq(CS, M)})$$

  where $PS1$ may be the same as $PS2$ and enq is a standard function for queues, taking a queue q and an element e and putting e into q at bottom.

- Receipt of Message is described as the following transition rule:

  $$(\text{p-state}[P] : PS1)(\text{c-state}[Q, P, N] : M \mid \text{CS}) \Rightarrow$$
  $$(\text{p-state}[P] : PS2)(\text{c-state}[Q, P, N] : \text{CS})$$

  where $PS1$ may be the same as $PS2$. The operator $\_|\_$ is used to construct queues of messages. For $m \in$ Msg and $q \in$ MsgQueue, $m \mid q \in$ MsgQueue, where m is the top message of the queue.

Although three kinds of actions are generally described by only three transition rules, there may be more than three ground instances of the transition rules. This is caused by the number of states for each process, the number of channels, the number of states for each channel, etc. Given a transition rule L $\Rightarrow$ R, a ground instance of the transition rule is obtained by replacing each variable in L $\Rightarrow$ R with a ground constructor term (or a value) of the sort of the variable.

Definition 2 (TR_{UDS})**.** Let TR_{UDS} be the set of all ground instances of the three transition rules.

3.1.3 State Machine M_{UDS}

A UDS is formalized as state machine $M_{UDS} \triangleq \langle S_U DS, I_{UDS}, T_{UDS} \rangle$. $S_U DS$ is the set of all ground constructor terms whose sorts are ConFigure. I_{UDS} is a subset of $S_U DS$ such that for each state $s \in I_{UDS}$, for each channel c in s the message queue in c is empty, for each process $p \in$ Pid there exists at most one p-state[p] observable component in s, for each (p, q, n) where $p, q \in$ Pid and $n \in$ Nat, there exists at most one c-state[p, q, n] observable component in s, and there is no dangling channel in s. T_{UDS} is the binary relation over $S_U DS$ made from TR_{UDS}.

Definition 3 (M_{UDS})**.** The state machine formalizing a UDS is $M_{UDS} \triangleq \langle S_U DS, I_{UDS}, T_{UDS} \rangle$, where

1. $S_U DS$ is the set of all ground constructor terms whose sorts are Config;

2. I_{UDS} is a subset of S_UDS such that $(\forall s \in I_{UDS})\,(\forall c \in \text{chans}(s))\,(\text{msg}(c) = \text{empChan})$, $(\forall s \in I_{UDS})\,(\forall p \in \text{Pid})\,(\#(s,p) \leq 1), (\forall s \in I_{UDS})\,(\forall p, q \in \text{Pid})$ $(\forall n \in \text{Nat})\,(\#(s,p,q,n) \leq 1)$ and $(\forall s \in I_{UDS})\,(\forall(\text{c-state}[p,q,n] : \text{cs}) \in s)$ $((\#(s,p) = 1) \wedge (\#(s,q) = 1))$;
3. T_{UDS} is the binary relation over S_UDS defined as follows: $\{(C, C') \mid C,\ C' \in \text{Config},\ L \Rightarrow R \in TR_{UDS},\ \exists C''.(C = LC'' \wedge C' = RC'')\}$.

Function chans gets all channels of a state $s \in S_UDS$, msg gets the state of a channel c in s, and # counts the number of occurrences of a process or a channel in s.

3.2 Generating State Machine M_{CLDSA} from State Machine M_{UDS}

Let M_{CLDSA} be the state machine that formalizes the UDS-CLDSA. Because CLDSA runs concurrently with, but does not alter, the behaviors of a UDS, it is possible to generate the formalization of the UDS-CLDSA from the UDS's. The authors in [18] show their way to do that. Carefully investigating, however, we found mistakes in their formalizations. Our detection of their mistakes will be presented in the rest of this section, where we focus on how to generate M_{CLDSA} from M_{UDS}. Let us consider the state expression and the state transition for M_{CLDSA}.

3.2.1 State Expression

Each state of M_{CLDSA} should consist of the local states of all processes and channels, the state (called the start state) when CLDSA initiates, the snapshot, the state (called the finish state) when CLDSA terminates and the information to control the behaviors of CLDSA.

With superficially observing, we may mistakenly comprehend that the local states of each process and channel in the UDS-CLDSA are exactly the same as the one of a UDS. They are, however, different. The key difference is that due to the working of CLDSA, a channel includes not only the normal data messages as those of a UDS, but also the control messages, markers. This is an important point needed to be noticed explicitly when modeling the system. As mistakes, the authors in [18] did not see this point. They consider the states of a channel in the system exactly the same as those for a UDS. The same sorts Msg, MsgQueue, OCom and Config for messages, the sequence of messages, observable components and a soup of *p-state* and *c-state* observable components, respectively, are used in their formalization of the UDS-CLDSA. Overcoming the problem, we use other sorts to describe the UDS-CLDSA. In detail, MMsg, a super-sort of sort Msg, is used for messages and markers. The sorts BOCom and BConfig, replacing OCom and Config, correspondingly, are used for observable components and a soup of *p-state* and *c-state* observable components. For this end, the following sorts and operators are defined:

subsort Msg < MMsg.

op marker :→ MMsg[*ctor*].

op empChan :→ MMsgQueue[*ctor*].

op _|_ : MMsg MMsgQueue → MMsgQueue[*ctor*].

op p-state[_]:_ : Pid PState → BOCom[*ctor*].

op c-state[_, _, _]:_ : Pid Pid Nat MMsgQueue → BOCom[*ctor*].

subsort BOCom < BConfig.

op empBConfig :→ BConfig[*ctor*].

op _ _ : BConfig BConfig → BConfig[ctor assoc comm id:empBConfig].

We use base-state(bc), start-state(sc), snapshot(©ssc) and finish-state(fc) called meta configuration components, in which bc, sc, ssc and fc are ground constructor terms of sort BConfig, to express the local states of all processes and channels, the start state, the snapshot and the finish state, respectively. The corresponding sort for those components is MBCom. The information to control behaviors of CLDSA is expressed as control(ctl) that is also a meta configuration component. ctl is a soup of cnt, prog, #ms, and done control observable components that will be described soon. CtlOCom is the sort for those components, and CtlConfig is the sort for soups of CtlOCom.

ops base-state start-state snapshot finish-state : BConfig→ MBCom[ctor].

op control : CtlConfig → MBCom[ctor].

The control observable components are as follows.

(cnt : n): n is the number of processes that have not yet completed CLDSA.

(prog[p] : pg): pg is the progress (notYet, started or completed) of a process p, indicating that the process has not yet started, has started, or completed CLDSA.

(#ms[p] : n): n is the number of incoming channels to a process p from which markers have not yet been received.

(done[p, q, n] : b): b is either $true$ or $false$. If b is $true$, q has received a marker from the incoming channel identified by n from p, otherwise, q has not.

Each state of M_{CLDSA} is expressed as the soup of the meta configuration components, which is in the form:

base-state(bc) start-state(sc) snapshot(©ssc) finish-state(fc) control(ctl)

which is called a meta configuration and the corresponding sort is MBConFigure. We define the following operators for the sort:

subsort MBCom < MBConfig.

op _ _ : MBConfig MBConfig → MBConfig[ctor assoc comm].

Initially, *bc* is an initial state of M_{UDS}, and all of *sc*, *ssc* and *fc* are *empBConfig*. The number of processes that have not yet completed CLDSA is equal to the number of processes in the system, the progress of all processes are *notYet*, and all processes have not yet received any markers. If *fc* is not *empBConfig*, a distributed snapshot has been taken and then *ssc* is the snapshot.

3.2.2 State Transitions

Each process in the system preserves the basic actions of those for a UDS, but needs to do two more kinds of actions for executing CLDSA as follows.

iv. Record of Process State: it may record its state and put markers into all of its outgoing channels when it has not yet received any markers, and

v. Receipt of Marker: it may get a marker from one of its incoming channels.

In the following part, P, $Q \in$ Pid, $PS, PS1, PS2 \in$ PState, BC, $SSC \in$ BConfig, $MMS \in$ MMsgQueue, $CC \in$ CtlConfig, $N \in$ Nat, $NzN \in$ NzNat and $M \in$ Msg are variables of those sorts, where NzNat is the sort for non-zero natural numbers and a subsort of Nat.

- Change of Process State is described as the following transition rule:

  base-state((p-state[P] : $PS1$)BC) ⇒ base-state((p-state[P] : $PS2$)BC).

- Sending of Message is described as the following transition rule:

  base-state((p-state[P] : $PS1$) (c-state[P, Q, N] : MMS)BC) ⇒
  base-state((p-state[P] : $PS2$) (c-state[P, Q, N] : enq(MMS, M))BC).

- Receipt of Message is split into four subcases:

  1. The process has not yet started CLDSA.

     base-state((p-state[P] : $PS1$) (c-state[Q, P, N] : $M \mid MMS$)BC)
     control((prog[P] : notYet) CC)
     ⇒
     base-state((p-state[P] : $PS2$) (c-state[Q, P, N] : MMS)BC)
     control((prog[P] : notYet)CC).

  2. The process has completed CLDSA.

     base-state((p-state[P] : $PS1$) (c-state[Q, P, N] : $M \mid MMS$)BC)
     control((prog[P] : completed)CC)
     ⇒
     base-state((p-state[P] : $PS2$) (c-state[Q, P, N] : MMS)BC)
     control((prog[P] : completed)CC).

3. The process has started CLDSA, not yet completed it, and not yet received a marker from the incoming channel.

   base-state((p-state[P] : $PS1$) (c-state[Q, P, N] : M | *MMS*)BC)
   snapshot((c-state[Q, P, N] : *MMS'*)SSC)
   control((prog[P] : started)(done[Q, P, N] : *false*)CC)
   $\Rightarrow$
   base-state((p-state[P] : $PS2$)(c-state[Q, P, N] : *MMS*)BC)
   snapshot((c-state[Q, P, N] : enq(*MMS'*, M))SSC)
   control((prog[P] : started)(done[Q, P, N] : *false*)CC).

4. The process has started CLDSA, but not yet completed it and it has already received a marker from the incoming channel.

   base-state((p-state[P] : $PS1$)(c-state[Q, P, N] : M | *MMS*)BC)
   control((prog[P] : started)(done[Q, P, N] : *true*)CC)
   $\Rightarrow$
   base-state((p-state[P] : $PS2$)(c-state[Q, P, N] : *MMS*)BC)
   control((prog[P] : started)(done[Q, P, N] : *true*)CC).

- Record of Process State is split into two subcases:

1. The process globally initiates CLDSA. This case is further split into three subcases:

   (a) The UDS only consists of the process.

   base-state((p-state[P] : PS)) start-state(empBConfig)
   snapshot(empBConfig) finish-state(empBConfig)
   control((prog[P] : notYet)(cnt : 1)(#ms[P] : 0)CC)
   $\Rightarrow$
   base-state((p-state[P] : PS)) start-state((p-state[P] : PS))
   snapshot((p-state[P] : PS)) finish-state((p-state[P] : PS))
   control((prog[P] : completed)(cnt : 0)(#ms[P] : 0)CC).

   (b) The system consists of more than one process, and the process does not have any incoming channels.

   base-state((p-state[P] : PS)BC) start-state(empBConfig)
   snapshot(empBConfig) finish-state(empBConfig)
   control((prog[P] : notYet)(cnt : NzN)(#ms[P] : 0)CC)
   $\Rightarrow$
   base-state((p-state[P] : PS) bcast(BC, P, marker))
   start-state((p-state[P] : PS)BC) snapshot((p-state[P] : PS))
   control((prog[P] : completed)(cnt : sd(NzN, 1))(#ms[P] : 0)CC)x
   if NzN > 1

   where bcast is a function putting markers in all outgoing channels from process P and sd is a function for natural number taking two natural numbers x and y and then returning $x - y$ if $x > y$ and $y - x$ otherwise.

(c) The system consists of more than one process, and the process has one or more incoming channels.

base-state((p-state[P] : PS)BC) start-state(empBConfig)
snapshot(empBConfig) finish-state(empBConfig)
control((prog[P] : notYet)(#ms[P] : NzN')CC)
$\Rightarrow$
base-state((p-state[P] : PS) bcast(BC, P, marker))
start-state((p-state[P] : PS)BC)
snapshot((p-state[P] : PS) inchans(BC, P))
control((prog[P] : started)(#ms[P] : NzN')CC).

2. The process does not globally initiate CLDSA. This case is further split into three subcases:

(a) The process does not have any incoming channel, and there are no processes except for the process that have not completed CLDSA.

base-state((p-state[P] : PS)BC) start-state(SC) snapshot(SSC)
finish-state(empBConfig)
control((prog[P] : notYet)(cnt : 1)(#ms[P] : 0)CC)
$\Rightarrow$
base-state((p-state[P] : PS)) start-state(SC)
snapshot((p-state[P] : PS)SSC)
finish-state((p-state[P] : PS)BC)
control((prog[P] : completed)(cnt : 0)(#ms[P] : 0)CC)
if ($SC \neq$ empBConfig).

(b) The process does not have any incoming channels, and there are some other processes that have not completed CLDSA.

base-state((p-state[P] : PS)BC) start-state(SC) snapshot(SSC)
control((prog[P] : notYet)(cnt : NzN)(#ms[P] : 0)CC)
$\Rightarrow$
base-state((p-state[P] : PS) bcast(BC, P, marker))
start-state(SC) snapshot((p-state[P] : PS)SSC)
control((prog[P] : completed)(cnt : sd(NzN, 1))(#ms[P] : 0)CC)
if ($SC \neq$ empBConfig)$\wedge$(NzN > 1).

(c) The process has some incoming channels.

base-state((p-state[P] : PS)BC) start-state(SC) snapshot(SSC)
control((prog[P] : notYet)(#ms[P] : NzN')CC)
$\Rightarrow$
base-state((p-state[P] : PS) bcast(BC, P,marker))
start-state(SC) snapshot((p-state[P] : PS) inchans(BC, P)SSC)
control((prog[P] : started)(#ms[P] : NzN')CC)
if ($SC \neq$ empBConfig).

- Receipt of Marker is split into two subcases:

  1. The process has not yet started CLDSA. This case is further split into three subcases:

     (a) The process has only one incoming channel, and there are no processes that have not yet completed CLDSA except for the process.

     base-state((p-state[P] : PS)(c-state[Q, P, N] : marker | *MMS*)BC)
     snapshot(SSC) finish-state(empBConfig)
     control((prog[P] : notYet)(cnt : 1)(#ms[P] : 1)
     (done[Q, P, N] : *false*)CC)
     $\Rightarrow$
     base-state((p-state[P] : PS)(c-state[Q, P, N] : *MMS*)BC)
     snapshot((p-state[P] : PS)(c-state[Q, P, N] : empChan)SSC)
     finish-state((p-state[P] : PS)(c-state[Q, P, N] : *MMS*)BC)
     control((prog[P] : completed)(cnt : 0)(#ms[P] : 0)(done[Q, P, N] : *true*)CC).

     (b) The process has only one incoming channel, and there are some other processes that have not yet completed CLDSA.

     base-state((p-state[P] : PS)(c-state[Q, P, N] : marker | *MMS*)BC)
     snapshot(SSC) control((prog[P] : notYet)(cnt : NzN)(#ms[P] : 1)
     (done[Q, P, N] : *false*)CC)
     $\Rightarrow$
     base-state((p-state[P] : PS)(c-state[Q, P, N] : *MMS*)
     bcast(BC, P,maker)) snapshot((p-state[P] : PS)(c-state[Q, P, N] : empChan)SSC)
     control((prog[P] : completed)(cnt : sd(NzN, 1))(#ms[P] : 0)
     (done[Q, P, N] : *true*)CC)
     if NzN $>$ 1.

     (c) The process has more than one incoming channels.

     base-state((p-state[P] : PS)(c-state[Q, P, N] : marker | *MMS*)BC)
     snapshot(SSC) control((prog[P] : notYet)(cnt : NzN)
     (#ms[P] : NzN')(done[Q, P, N] : *false*)CC)
     $\Rightarrow$
     base-state((p-state[P] : PS)(c-state[Q, P, N] : *MMS*)
     bcast(BC, P,maker)) snapshot((p-state[P] : PS)
     (c-state[Q, P, N] : empChan)inchans(BC, P)SSC)
     control((prog[P] : started)(cnt : sd(NzN, 1))(#ms[P] : sd(NzN', 1))
     (done[Q, P, N] : *true*)CC)
     if NzN' $>$ 1.

  2. The process has already started CLDSA. This case is further split into three subcases:

(a) There is no incoming channel from which markers have not been received except for the incoming channel, and there are no processes that have not yet completed CLDSA except for the process.

base-state((p-state[P] : PS)(c-state[Q, P, N] : marker | *MMS*)BC)
finish-state(empBConfig) control((prog[P] : started)(cnt : 1)
(#ms[P] : 1)(done[Q, P, N] : *false*)CC)
$\Rightarrow$
base-state((p-state[P] : PS)(c-state[Q, P, N] : *MMS*)BC)
finish-state((p-state[P] : PS)(c-state[Q, P, N] : *MMS*)BC)
control((prog[P] : completed)(cnt : 0)(#ms[P] : 0)
(done[Q, P, N] : *true*)CC).

(b) There are no incoming channels from which markers have not been received except for the incoming channel, and there are some other processes that have not yet completed CLDSA.

base-state((p-state[P] : PS)(c-state[Q, P, N] : marker | *MMS*)BC)
control((prog[P] : started)(cnt : NzN)(#ms[P] : 1)
(done[Q, P, N] : *false*)CC)
$\Rightarrow$
base-state((p-state[P] : PS)(c-state[Q, P, N] : *MMS*)BC)
control((prog[P] : completed)(cnt : sd(NzN, 1)(#ms[P] : 0)
(done[Q, P, N] : *true*)CC)
if NzN $>$ 1.

(c) There are some other incoming channels from which markers have not been received.

base-state((p-state[P] : PS)(c-state[Q, P, N] : marker | *MMS*)BC)
control((prog[P] : started)(cnt : NzN)(#ms[P] : NzN')
(done[Q, P, N] : *false*)CC)
$\Rightarrow$
base-state((p-state[P] : PS)(c-state[Q, P, N] : *MMS*)BC)
control((prog[P] : started)(cnt : NzN)(#ms[P] : sd(NzN', 1))
(done[Q, P, N] : *true*)CC) *[]* if NzN' $>$ 1.

There are 18 transition rules described as above. Those transition rules are classified into three parts: UDS, UDS & CLDSA, and CLDSA. The UDS part consists of the transition rules describing the actions purely related to the UDS, namely i, ii and iii-1. The UDS part depends on the UDS concerned, can be constructed from the three transition rules of the UDS and changes the base-state meta configuration component of a state of M_{CLDSA}. The UDS & CLDSA part also depends on the UDS concerned and can be constructed from the three transition rules of the UDS, but changes the other meta configuration components of a state of M_{CLDSA} as well. Three transition rules describing actions iii-2, iii-3 and iii-4 are in the UDS & CLDSA part. The CLDSA part is independent from the UDS concerned, can be constructed regardless of any UDSs, and does not change the base-state meta configuration component of a state of M_{CLDSA}. The transition rules describing two kinds of actions iv and v are in the CLDSA part.

Definition 4 (TR_{CLDSA})**.** Let TR_{CLDSA} be the set of all ground instances of the 18 transition rules.

3.2.3 State Machine M_{CLDSA}

We propose the function CL that takes a state machine M_{UDS} and returns another state machine M_{CLDSA}. Note that M_{UDS} is the state machine of a UDS and M_{CLDSA} is the state machine of the UDS-CLDSA. Since the authors in [18] treat the states of processes and channels of the UDS superimposed by CLDSA as the same as those of a UDS, the definition of function CL in [18] is incorrect. The definition is as follows:

> For a state machine $M_{UDS} \triangleq \langle S_U DS, I_{UDS}, T_{UDS} \rangle$, CL($S_U DS$) = {base-state($bs$) start-state($ss$) snapshot($sss$)© f-state(fs) control($ctl$) | $bs \in S_U DS$, $ss \in$ Config, $sss \in$ Config, $fs \in$ Config, $ctl \in$ CtlConfig}, where Config and CtlConfig are used as the sets of terms whose sorts are Config and CtlConfig, respectively.

They consider that bs is in $S_U DS$. Obviously, this is incorrect since the channels in bc may contain markers, which do not exist in the channels of a UDS. Hence, bs cannot be in $S_U DS$. We redefine the function CL as follows:

Definition 5 (CL(M_{UDS}))**.** For a state machine $M_{UDS} \triangleq \langle S_U DS, I_{UDS}, T_{UDS} \rangle$ formalizing a UDS, CL is the function that takes M_{UDS} and returns CL(M_{UDS}) $\triangleq$ $\langle$CL$_{State}(S_U DS)$,CL$_{Init}(I_{UDS})$,CL$_{Trans}(T_{UDS})\rangle$, where

1. CL$_{State}(S_U DS)$ is the set of all ground constructor terms of sort MBConfig;
2. CL$_{Init}(I_{UDS})$ is {base-state(bc) start-state(empBConfig) snapshot(©empBConfig) finish-state(empBConfig) control(ctl) | bc $\in I_{UDS}$, ctl = InitCtlConfig(bc)};
3. CL$_{Trans}(T_{UDS}) \subseteq$ CL$_{State}(S_U DS) \times$ CL$_{State}(S_U DS)$ is $\{(MC, MC') \mid MC, MC' \in \text{MBConfig}, L \Rightarrow R \in TR_{CLDSA}, \exists MC''.(MC = L MC'' \wedge MC' = R MC'')\}$.

Function InitCtlConfig(bc) initializes values for all control information components. Let M_{CLDSA} be CL(M_{UDS}). Note that S_{CLDSA} is the set of all ground constructor terms of sort MBConfig, although each reachable state from an initial state in S_{CLDSA} is in the following form:

base-state(bc) start-state(sc) snapshot(ssc) finish-state(fc) control(ctl).

Some functions on S_{CLDSA} are defined for convenience.

Definition 6 (b-state, s-state, snapshot, f-state, finished)**.** For each $s \in S_{CLDSA}$,

- b-state(s) is bc if there exists exactly one occurrence of the base-state(bc) meta configuration component in s and empBConfig otherwise,
- s-state(s) is sc if there exists exactly one occurrence of the start-state(sc) meta configuration component in s and empBConfig otherwise,
- snapshot(s) is ssc if there exists exactly one occurrence of the snapshot(ssc) meta configuration component in s and empBConfig otherwise,
- f-state(s) is fc if there exists exactly one occurrence of the finish-state(fc) meta configuration component in s and empBConfig otherwise, and
- finished(s) is false if f-state(s) is empBConfig and *true* otherwise.

The following is the definition that CLDSA has terminated in a state s in M_{CLDSA}:

Definition 7 ($M_{CLDSA} \models$ terminated(s))**.** For a state machine $M_{UDS} \triangleq \langle S_U DS, I_{UDS}, T_{UDS} \rangle$, for each $s \in S_{CLDSA}$, $M_{CLDSA} \models$ terminated(s) if and only if finished(s).

In the rest of the paper, terminated is abbreviated as trmtd. We have the following proposition on M_{CLDSA}:

Proposition 1 (No marker in s-state, snapshot and f-state)**.** For each $s \in S_{CLDSA}$, if $M_{CLDSA} \models$ trmtd(s), then there is no marker in s-state(s), snapshot(s) and f-state(s), equivalently that the least sort of s-state(s), snapshot(s) and f-state(s) are ConFigure.

Note that, whenever CLDSA has terminated in a state s, the function s-state(s), snapshot(s) and f-state(s) return the start state, the snapshot and the finish state, respectively.

4 A MORE FAITHFUL DEFINITION OF THE DSR PROPERTY

4.1 The Informal Description of the DSR Property

The informal description of the DSR Property is given in [13] as follows. Let s_1, s_* and s_2 be the state in which CLDSA initiates, the snapshot taken, and the state in which CLDSA terminates, respectively. Although the snapshot s_* may not be identical to any of the global states that occur in the computation from s_1 to s_2, one desired property (called the DSR property) CLDSA should satisfy is that s_* is reachable from s_1 and s_2 is reachable from s_*, whenever CLDSA terminates. Note that s_1, s_2 and s_* are states of the UDS, but not those of the UDS-CLDSA.

4.2 Formal Definition of the DSR Property

Infinite sequences of states called paths are generated from a state machine because T is total. Paths are defined as follows.

Definition 8 (Path)**.** A path π of a state machine $M \triangleq \langle S, I, T\rangle$ from a state s_0 is an infinite sequence of states $\pi \triangleq (s_0, s_1, s_2, \dots)$, where $(\forall i \geq 0)((s_i, s_{i+1}) \in T)$. π_i denotes the i^{th} state (i.e., s_i) in π and Π denotes the set of all paths of M.

For a state machine M, $M, \pi \models \text{isReachable}(s_2, s_1)$ if and only if s_2 is reachable from s_1 in a path π in M and then $M \models \text{isReachable}(s_2, s_1)$ if and only if s_2 is reachable from s_1 in M.

Definition 9 (Reachabilty in M)**.** For a state machine $M \triangleq \langle S, I, T\rangle$, for each $\pi \in \Pi$ and each $s_1, s_2 \in S$, $M, \pi \models \text{isReachable}(s_2, s_1)$ if and only if $(\exists i, j \in Nat)\,(i \leq j \wedge s_1 = \pi_i \wedge s_2 = \pi_j)$, and $M \models \text{isReachable}(s_2, s_1)$ if and only if $(\exists \pi \in \Pi)\,(M, \pi \models \text{isReachable}(s_2, s_1))$.

In other words, a state s_2 is said to be reachable from a state s_1 if and only if s_1 can go to s_2 by zero or more state transition steps in the state machine M.

In the informal description of the DSR property, it is checked that CLDSA terminates, and it is checked that some states of a UDS are reachable from some others in the UDS but not the UDS-CLDSA. Accordingly, the property involves two systems, a UDS and the UDS-CLDSA, and hence we need to use two state machines M_{UDS} and M_{CLDSA} to faithfully define the DSR property. Our definition of the DSR property is as follows.

Definition 10 (The DSR Property)**.** For a state machine $M_{UDS} \triangleq \langle S_UDS, I_{UDS}, T_{UDS}\rangle$, $(\forall s \in S_{CLDSA})\,(M_{CLDSA} \models \text{trmtd}(s) \Rightarrow M_{UDS} \models \text{isReachable}(s_*, s_1) \wedge M_{UDS} \models \text{isReachable}(s_2, s_*))$, where $s_1 = \text{s-state}(s)$, $s_* = \text{snapshot}(s)$ and $s_2 = \text{f-state}(s)$.

5 THE THEOREM ON EQUIVALENCE OF THE TWO DEFINITIONS OF THE DSR PROPERTY

Since our new definition of the DSR property is more likely to faithfully express the informal description of the property, it can be used to more faithfully model check that CLDSA enjoys the property. However, due to involving two state machines, it is not straightforward to directly model check the new definition with an existing model checker. This is because existing temporal logics, such as LTL and CTL, used for model checking, only consider one state machine, more precisely one Kripke structure. Because the existing definition of the DSR property has been model checked in [16], the equivalence of the new definition and the existing one guarantees that we can use the existing model checking approach to model checking for the new definition. Therefore, we prove a theorem saying that our new definition is equivalent to the existing definition, which also confirms the validity of the existing model checking approach.

Although the DSR property is encoded in terms of the Maude search command in the existing study, the existing definition can be represented in terms of state machines. Let us suppose that there are n processes in a UDS and let $p_1, \dots, p_n$

be their identifications, namely that Pid is $\{p_1, \dots, p_n\}$, where $n \geq 1$. Let ctl be (prog$[p_1]$: notYet)…(prog$[p_n]$: notYet) in the rest of the paper. The existing definition of the DSR property is represented in terms of state machines as follows:

> For a state machine $M_{UDS} \triangleq \langle S_U DS, I_{UDS}, T_{UDS} \rangle$, $(\forall s \in S_{CLDSA})\,(M_{CLDSA} \models \text{trmtd}(s) \Rightarrow M_{CLDSA} \models \text{isReachable}(\text{base-state}(s_*)\,\text{control}(\text{ctl}), \text{base-state}(s_1)\,\text{control}(\text{ctl})) \wedge M_{CLDSA} \models \text{isReachable}(\text{base-state}(s_2)\,\text{control}(\text{ctl}), \text{base-state}(s_*)\,\text{control}(\text{ctl})))$, where $s_1 = \text{s-state}(s)$, $s_* = \text{snapshot}(s)$ and $s_2 = \text{f-state}(s)$.

Both of the definitions are checking the termination of CLDSA in M_{CLDSA}. However, the reachability is checked in the other state machine M_{UDS} in the new definition, while it is checked in the same state machine M_{CLDSA} in the existing definition. This is the key difference between the two definitions. Although the two definitions are seemingly different, we realize that the new one coincides with the existing one [16]. Hence we prove the following theorem saying that two definitions are equivalent.

Theorem 1 (Equivalence of the Two Definitions)**.** For a state machine $M_{UDS} \triangleq \langle S_U DS, I_{UDS}, T_{UDS} \rangle$, $(\forall s \in S_{CLDSA})\,(M_{CLDSA} \models \text{trmtd}(s) \Rightarrow M_{UDS} \models \text{isReachable}(s_*, s_1) \wedge M_{UDS} \models \text{isReachable}(s_2, s_*)) \Leftrightarrow (M_{CLDSA} \models \text{trmtd}(s) \Rightarrow M_{CLDSA} \models \text{isReachable}(\text{base-state}(s_*)\,\text{control}(\text{ctl}), \text{base-state}(s_1)\,\text{control}(\text{ctl})) \wedge M_{CLDSA} \models \text{isReachable}(\text{base-state}(s_2)\,\text{control}(\text{ctl}), \text{base-state}(s_*)\,\text{control}(\text{ctl})))$, where $s_1 = \text{s-state}(s)$, $s_* = \text{snapshot}(s)$ and $s_2 = \text{f-state}(s)$.

The only difference between the new definition and the existing one is the conclusion part of the implications, in which the different state machines are used to check the reachability in each definition. If we can prove that the conclusion parts are equivalent, then the two definitions are equivalent. The equivalence of the conclusion parts means that reachability is preserved between M_{UDS} and M_{CLDSA}. Therefore, to prove Theorem 1, we prove Lemma 1 on reachability preservation. Lemma 1 asserts that reachability is preserved between M_{UDS} and M_{CLDSA}. The lemma is as follows.

Lemma 1 (Reachability Preservation)**.** For a state machine $M_{UDS} \triangleq \langle S_U DS, I_{UDS}, T_{UDS} \rangle$, $(\forall s_1, s_2 \in S_U DS)\,(M_{UDS} \models \text{isReachable}(s_2, s_1) \Leftrightarrow M_{CLDSA} \models \text{isReachable}(\text{base-state}(s_2)\,\text{control}(\text{ctl}), \text{base-state}(s_1)\,\text{control}(\text{ctl})))$.

We first prove as Lemma 2 and Lemma 3 that one-step reachability is preserved between M_{UDS} and M_{CLDSA} to prove Lemma 1. The two lemmas are as follows.

Lemma 2 (One-step Reachability Preservation from M_{UDS} to M_{CLDSA})**.** $\forall s_1, s_2 \in S_U DS$ such that s_1 goes to s_2 with one state transition step in M_{UDS}, base-state(s_1) control(ctl) goes to base-state(s_2) control(ctl) with one state transition step in M_{CLDSA}.

Lemma 3 (One-step Reachability Preservation from M_{CLDSA} to M_{UDS})**.** $\forall\ s_1,\ s_2 \in S_UDS$ such that base-state(s_1) control(ctl) goes to base-state(s_2) control(ctl) with one state transition step in M_{CLDSA}, s_1 goes to s_2 with one state transition step in M_{UDS}.

For each UDS, T_{UDS} is constructed from the three transition rules and T_{CLDSA} is constructed from the 18 transition rules. Therefore, all we have to do is to take into account the three transition rules and the 18 transition rules to discuss T_{UDS} and T_{CLDSA}, respectively. In the following proofs, $p, q \in$ Pid, $ps_1, ps_2 \in$ PState, $cs \in$ MsgQueue, $m \in$ Msg, $bc \in$ Config and $n \in$ Nat are fresh constants of those sorts.

Proof. (Proof Sketch of Lemma 2.) Assume that s_1 goes to s_2 by a state transition t in M_{UDS}. Our proof shows that there exists a state transition t' in M_{CLDSA} that moves base-state(s_1) control(ctl) to base-state(s_2) control(ctl). Let us consider the case in which t is constructed from the transition rule that describes Sending of Message in M_{UDS}. It suffices to consider s_1 as an arbitrary state (p-state[p] : ps_1) (c-state[p, q, n] : cs)bc in S_UDS to which the transition rule can be applied. Therefore, s_2 is (p-state[p] : ps_2) (c-state[p, q, n] : enq(cs, m))bc. Then, base-state(s_1) control(ctl) is base-state((p-state[p] : ps_1)(c-state[p, q, n] : cs)bc) control(ctl), and base-state(s_2) control(ctl) is base-state((p-state[p] : ps_2) (c-state[p, q, n] : enq(cs, m)) bc) control(ctl). The transition rule that describes Sending of Message in M_{CLDSA} can be applied to base-state(s_1) control(ctl) and obtains base-state(s_2) control(ctl). Hence, there exists t'. The case has been discharged. We can deal with the other two cases that correspond to Change of Process State and Receipt of Message, respectively. □

Proof. (Proof Sketch of Lemma 3.) Assume that base-state(s_1) control(ctl) goes to base-state(s_2) control(ctl) by a state transition t in M_{CLDSA}. Because $s_1 \in S_UDS$, there is no marker in s_1. Moreover, ctl is (prog[p_1] : notYet) . . . (prog[p_n] : notYet). This is why any of the transition rules that describe Record of Process State and Receipt of Marker in M_{CLDSA} cannot be applied to base-state(s_1) control(ctl). Therefore, t is not a state transition constructed from those transition rules. Any of the transition rules that describe the 2nd, 3rd and 4th sub-cases of Receipt of Message in M_{CLDSA} cannot be applied to base-state(s_1) control(ctl), neither. Therefore, t is not a state transition constructed from those transition rules, neither. Then, all we have to do is to consider the transition rules that describe Change of Process State, Sending of Message and the 1st part of Receipt of Message in M_{CLDSA}. The same proof strategy used in the proof of Lemma 2 can be used to show that there exists a state transition that moves s_1 to s_2 in M_{UDS} for each state transition that moves base-state(s_1) control(ctl) to base-state(s_2) control(ctl) in M_{CLDSA}. □

Proof. (Proof of the "if" part of Lemma 1.) We prove that $\forall s_1, s_2 \in S_UDS$, if $M_{CLDSA} \models$ isReachable(base-state(s_2) control(ctl), base-state(s_1) control(ctl)), then $M_{UDS} \models$ isReachable(s_2, s_1).

Assume that $M_{CLDSA} \models$ isReachable(base-state(s_2) control(ctl), base-state(s_1) control(ctl)) and then there must be a natural number k such that base-state(s_1) control(ctl) goes to base-state(s_2) control(ctl) by k state transition steps in M_{CLDSA}. The proof is done by induction on k.

Base case: Since base-state(s_1) control(ctl) is the same as base-state(s_2) control(ctl) in this case, s_1 is the same as s_2. So, this case is discharged.

Induction case: Suppose that base-state(s_1) control(ctl) moves to base-state(s_2) control(ctl) by $k+1$ transition steps and the $k+1$ transitions taken are $t_1, \ldots, t_{k+1}$. As shown in Figure 5, base-state(s') control(ctl) is the state to which base-state(s_1) control(ctl) moves by the first k transition steps, namely that $M_{CLDSA} \models$ isReachable(base-state(s') control(ctl), base-state(s_1) control(ctl)). From the induction hypothesis, $M_{UDS} \models$ isReachable(s', s_1). Since base-state(s') control(ctl) moves to base-state(s_2) control(ctl) by one transition step in M_{CLDSA}, s' also moves to s_2 by one transition step in M_{UDS} from Lemma 3. Then, this case is also discharged. Figure 5 shows the correspondence between the transitions in M_{CLDSA} and M_{UDS}.

□

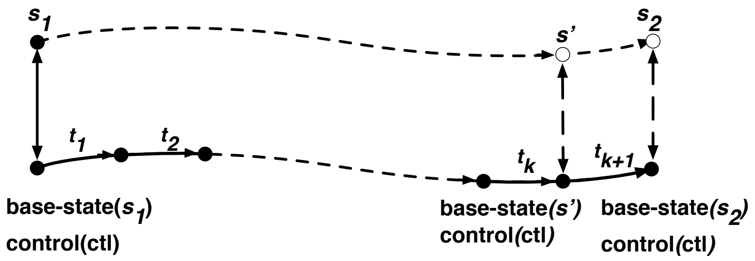

s1
s'
s2
t1
t2
tk
tk+1
base-state(s1) control(ctl)
base-state(s') control(ctl)
base-state(s2) control(ctl)

Figure 3. The correspondence between the transitions in M_{CLDSA} and M_{UDS}

Proof. (Proof Sketch of the "only if" part of Lemma 1.) We prove that $\forall s_1, s_2 \in S_UDS$, if $M_{UDS} \models$ isReachable(s_2, s_1), then ($M_{CLDSA} \models$ isReachable(base-state(s_2) control(ctl), base-state(s_1) control(ctl)).

Assume that $M_{UDS} \models$ isReachable(s_2, s_1) and then there must exist a natural number k such that s_1 goes to s_2 by k state transition steps in M_{UDS}. The proof is done by induction on k. Note that Lemma 2 is used in this proof. □

Proof. (Proof of Theorem 1.) Proof of Theorem 1 follows from Proposition 1 and Lemma 1. □

6 CLDSA DOES NOT ALTER THE BEHAVIORS OF A UDS

As control algorithms [21], DSAs should run concurrently but not interfere with the behaviors of a UDS. The behaviors of the algorithms are transparent to a UDS. It is necessary to prove that CLDSA does not alter the behaviors of a UDS to guarantee the correctness of the algorithm. We will prove that any original actions of each process of a UDS, namely sending a message, receiving a message and changing its state, are preserved by CLDSA.

If we prove that M_{CLDSA} simulates M_{UDS} and vice versa, we can state that the behaviors of a UDS are preserved by CLDSA. We propose a binary relation **r** between M_{UDS} and M_{CLDSA}, and then prove Theorem 1 saying that **r** is a bi-simulation relation between M_{UDS} and M_{CLDSA}. In the following part, for all states s, s' in state machine M, $s \rightsquigarrow_M s'$ denotes that state s moves to states s' by one state transition of M, and $s \rightsquigarrow_M^* s'$ denotes that state s moves to state s' by zero or more state transitions of M. Simulation from one state machine to another is defined as follows:

Definition 11 (Simulation from M_A to M_B)**.** Given two state machines $M_A \triangleq \langle S_A, I_A, T_A\rangle$ and $M_B \triangleq \langle S_B, I_B, T_B\rangle$, $r : S_A\ S_B \to$ Bool is called a *simulation* from M_A to M_B if it satisfies the following conditions:

1. For each $s_A \in I_A$ there exists $s_B \in I_B$ such that $r(s_A, s_B)$.
2. For each $s_A, s'_A \in S_A$ and $s_B \in S_B$ such that $r(s_A, s_B)$ and $s_A \rightsquigarrow_{M_A} s'_A$, there exists $s'_B \in S_B$ such that $r(s'_A, s'_B)$ and $s_B \rightsquigarrow_{M_B}^* s'_B$.

r is a bi-simulation if and only if it is a simulation from M_A to M_B and vice versa.

We recognize that with the exception of putting markers into the channels of a UDS, the algorithm does not change any original behavior of processes in the system. We propose a binary relation **r** between M_{UDS} and M_{CLDSA} saying that for each $s_1 \in S_UDS$ and each $s_2 \in S_{CLDSA}$, $\mathbf{r}(s_1, s_2)$ if and only if s_1 is the same as the state obtained by deleting all markers from s_2. The functions to delete all markers from one state of a UDS on which CLDSA is superimposed is implemented as the function delM as follows:

op delM: BConfig $\to$ Config.

eq delM(empBConfig) = empConfig.

eq delM((p-state[P] : PS) BCF) = (p-state[P] : PS)delM(BCF).

eq delM((c-state[P, Q, N] : MMS)BCF) =

(c-state[P, Q, N] : delMchan(MMS)) delM(BCF).

Where function delMchan deletes all markers in a sequence of messages.

The binary relation **r** is defined as follows:

Definition 12 (Binary relation **r**)**.** Given two state machines $M_{UDS} \triangleq \langle S_UDS, I_{UDS}, T_{UDS}\rangle$ and $M_{CLDSA} \triangleq \langle S_{CLDSA}, S_{CLDSA}, T_{CLDSA}\rangle, \forall s_{UDS} \in S_UDS$ and $\forall s_{CLDSA} \in S_{CLDSA}$, the binary relation $\mathbf{r} : S_UDSS_{CLDSA} \to$ Bool is defined as follows:

$$\mathbf{r}(s_{UDS}, s_{CLDSA}) \triangleq (s_{UDS} = delM(\text{b-state}(s_{CLDSA}))).$$

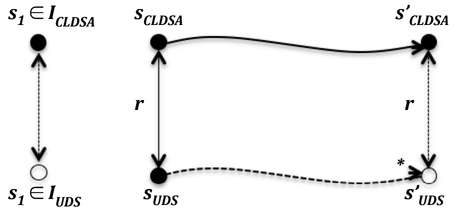


Figure 4. The binary relation $\mathbf{r}$ is a simulation from M_{CLDSA} to M_{UDS}

Theorem 2 (Bi-simulation relation $\mathbf{r}$)**.** Binary relation $\mathbf{r}$ is a bi-simulation relation between M_{UDS} and M_{CLDSA}.

We will prove that $\mathbf{r}$ is a simulation from M_{CLDSA} to M_{UDS} and vice versa. It suffices to only consider states reachable from the initial states in the proof. Initial states have a specific form of configuration and no observable component will be added to and/or deleted from initial states by any transition. The proof uses this fact.

Simulation from M_{CLDSA} to M_{UDS}. We will prove that $\mathbf{r}$ satisfies the following conditions. Figure 6 shows the diagrams corresponding to the two conditions.

Condition 1. For each $s_{CLDSA} \in S_{CLDSA}$ there exists $s_{UDS} \in I_{UDS}$ such that $\mathbf{r}(s_{UDS}, s_{CLDSA})$.

Proof. For each $s_{CLDSA} \in \mathrm{CL}_{Init}(I_{UDS})$, according to the definition of CL, s_{CLDSA} is in form of base-state(bc) start-state(empBConfig) snapshot(empBConfig) finsh-state(empBConfig) control(ctl), where $bc \in I_{UDS}$. Since b-state(s_{CLDSA}) $= bc$ and $bc \in I_{UDS}$, let us choose s_{UDS} is bc. Because delM(b-state(s_{CLDSA})) $= bc$, $s_{UDS} = delM(b - state(s_{CLDSA}))$. We have $\mathbf{r}(s_{UDS}, s_{CLDSA})$. This condition is satisfied. □

Condition 2. For each s_{CLDSA}, $s'_{CLDSA} \in S_{CLDSA}$ and $s_{UDS} \in S_UDS$ such that $\mathbf{r}(s_{UDS}, s_{CLDSA})$ and $s_{CLDSA} \rightsquigarrow_{M_{CLDSA}} s'_{CLDSA}$, there exists s'_{UDS} such that $\mathbf{r}(s'_{UDS}, s'_{CLDSA})$ and $s_{UDS} \rightsquigarrow^*_{M_{UDS}} s'_{UDS}$.

Proof. (Proof sketch.) The configuration of a state of M_{CLDSA} is as follows.

base-state(bc) start-state(sc) snapshot(ssc) finish-state(fc) control(ctl),

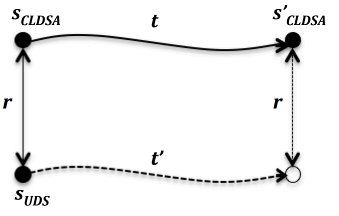
where $bc, sc, ssc, fc \in$ BConfig and ctl $\in$ CtlConFigure.

Because of $\mathbf{r}(s_{UDS}, s_{CLDSA})$, $s_{UDS} =$ delM(b-state(s_{CLDSA})) $=$ delM(bc). Let us assume that $s_{CLDSA} \rightsquigarrow_{M_{CLDSA}} s'_{CLDSA}$ by state transition t. The same as what we have mentioned above, we only take into account the three transition rules and the 18 transition rules to discuss T_{UDS} and T_{CLDSA}, respectively. Because the 18 transition rules are classified into three parts: UDS, UDS & CLDSA, and CLDSA, the state transitions can be also classified into the three parts. In what follows,

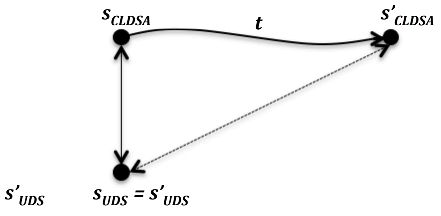
$p, q \in$ Pid, $ps_1, ps_2 \in$ PState, $cs \in$ MsgQueue, $m \in$ Msg, $bc, sc, ssc \in$ BConfig, $ctl \in$ CtlConfig and $n \in$ Nat are fresh constants of those sorts.

1. Let us consider the first case in which t is in the UDS part and the UDS & CLDSA part.

   Our proof shows that for any state transition t in the UDS and the UDS & CLDSA parts that moves s_{CLDSA} to s'_{CLDSA}, there exists s'_{UDS} to which s_{UDS} moves by one state transition such that $\mathbf{r}(s'_{CLDSA}, s'_{UDS})$ holds. We can find a state transition t' in M_{UDS} that can move $s_{UDS} =$ delM(b-state(s_{CLDSA})) to $s'_{UDS} =$ delM(b-state(s'_{CLDSA})). The existence of t' corresponding to t is shown in Figure 5 a).



a)



b)

Figure 5. Existing t' in M_{UDS} corresponding to t

   Let us consider the case in which t is constructed from the transition rule that describes Change of Process State in M_{CLDSA}. It suffices to consider s_{CLDSA} as base-state((p-state[p] : $ps1$)bc) start-state(sc) snapshot(ssc) finish-state(fc) control(ctl) to which the transition rule can be applied. Because of $\mathbf{r}(s_{UDS}, s_{CLDSA})$, $s_{UDS} =$ delM((p-state[p] : $ps1$)bc) = (p-state[p] : $ps1$) delM(bc) from the definition of function delM. Since s_{CLDSA} goes to s'_{CLDSA} by t, s'_{CLDSA} is base-state((p-state[p] : $ps2$)bc) start-state(sc) snapshot(ssc) finish-state(fc) control(ctl). Let t' be the state transition that is constructed from the transition rule that describes Change of Process State in M_{UDS}. Let s'_{UDS} be (p-state[p] : $ps2$) delM(bc). Then s_{UDS} can move to s'_{UDS} by t'. Because $s'_{UDS} =$ (p-state[p] : $ps2$) delM(bc) and delM(b-state(s'_{CLDSA})) = (p-state[p] : $ps2$) delM(bc), $s'_{UDS} =$ delM(b-state(s'_{CLDSA})). Therefore, $\mathbf{r}(s'_{UDS}, s'_{CLDSA})$ and $s_{UDS} \rightsquigarrow_{M_{UDS}} s'_{UDS}$ by t'. The case has been discharged. We can deal with the other two cases that correspond to Sending of Message and the 1st of Receipt of Message, respectively, likewise.

2. The last case in which t is constructed from the transition rule in the CLDSA part.

   Since CLDSA part does not change the base-state meta configuration component of a state of M_{CLDSA}. Our proof shows that we can choose as s'_{UDS} the same as s_{UDS} then s_{UDS} goes to s'_{UDS} by zero step and $\mathbf{r}(s'_{UDS}, s'_{CLDSA})$. This is shown in Figure 5 b).

From what have been proved above, we can see that relation $\mathbf{r}$ satisfies the two conditions of simulation from M_{CLDSA} to M_{UDS}. Therefore, $\mathbf{r}$ is a simulation relation from M_{CLDSA} to M_{UDS}. □

Simulation from M_{UDS} to M_{CLDSA} We will prove that $\mathbf{r}$ is a simulation from M_{UDS} to M_{CLDSA}.

Proof. (Proof Sketch.) Our proof shows that $\mathbf{r}$ satisfies the two conditions of simulation from M_{CLDSA} to M_{UDS}. It is straightforward to show that for each s in I_{UDS} there exists s' in I_{CLDSA} such that $\mathbf{r}(s, s')$ holds. To show the second condition, we need to consider the three (kinds of) transition rules of M_{UDS} and then it suffices to consider the rules of the UDS part and some rules of the UDS & CLDSA part in M_{CLDSA}. The proof can be conducted like we have done for the proof of simulation from M_{CLDSA} to M_{UDS}. □

7 RELATED WORK

Many researches [10, 11, 12] have been conducted to formally verify various distributed systems. Among them, [10] concentrates on model checking for distributed systems. The main contribution of the research is the design and implementation of the fair linear temporal logic of rewriting (LTLR) model checker, a model checker under localized fairness assumptions for Maude system. LTLR is an extension of LTL. So the Fair LTLR model checker is basically an extension of Maude LTL model checker dealing with fairness assumptions. Although the model checker tries to deal with several distributed algorithms, it has not yet considered DSAs. The authors in [11] deal with the problem of verification of asynchronous consensus algorithms, a fault-tolerant distributed algorithm. The challenge of the problem is that the state space is huge. Dealing with this problem, they have proposed a semi-automatic verification approach based on model checking technique. In their approach the problem of verification of asynchronous consensus algorithms is reduced to small model checking problems, namely the set of bounded model checking problems that can be solved efficiently by using bounded model checking with an SMT (Satisfiability Modulo Theories) solver. In detail, they adopt a round-based model called the Heard-Of (HO) model [20] to alleviate the problem. Their method can be used to model check several consensus algorithms up to around 10 processes. However, the method can only be applied to some consensus algorithms but not to DSAs.

CLDSA and its desired properties were initially introduced in [13]. In this, the DSR property is given in an informal way. Several studies are motivated by verification of snapshot algorithms. Among them, [7, 12] consider directly CLDSA. In [7], CLDSA is modelled in PROMELA, and then the model is simplified to be verifiable. However, only the UDS-CLDSA is modelled, and a property that is different from the DSR property is model checked for CLDSA. The authors in [12] focus on developing snapshot algorithms with formal proofs that guarantee the correctness of the algorithms. Some existing snapshot algorithms, such as CLDSA and Lai-Yang, are re-developed by using the Event B framework and refinement. Starting with a model providing an abstract view of a system and its behaviors, the model then is enriched more concretely by many refinement steps to derive the algorithms. To capture the complete and desired behaviors of snapshot algorithms, each refinement step must preserve essential desired properties ensuring a consistent cut. The properties are implemented as invariant conditions. This is also to ensure that the snapshot recorded by the deriving algorithm is consistent. Their experiments are conducted on fixed networks. Moreover, any of the properties they have considered are not necessarily the same as the DSR property.

8 CONCLUSION

The authenticity of a model checking relies on the faithfulness of the specifications of desired properties. It is expected that the desired properties are faithfully expressed in the specifications. Attempting to more faithfully model check the DSR property for CLDSA, we have given a more faithful formal definition of the DSR property. Our definition involves two state machines in which the termination is checked in the state machine M_{CLDSA} formalizing the UDS-CLDSA and the reachability is checked in the other state machine M_{UDS} formalizing a UDS. The checking of the reachability is different between the new definition and the existing definition. To guarantee that it suffices to model check the definition used in the existing study for CLDSA and the existing model checking approach can be used for this end, we have proved Theorem 1 saying that our formalization of the DSR property is equivalent to the existing one for each M_{UDS}. Moreover, we have proved Theorem 2 saying that M_{CLDSA} simulates M_{UDS} and vice versa to guarantee that CLDSA does not alter the behaviors of a UDS.

In the existing work [16], it is necessary to specify the UDS on which CLDSA is superimposed for each UDS in Maude to model check that the UDS on which CLDSA is superimposed enjoys the DSR property with the Maude search command. Moreover, the way to model check means to compare the numbers of solutions obtained by three search experiments. Therefore, it is not straightforward to construct a counterexample when the property is not fulfilled. The specification techniques described in the paper make it possible to specify CLDSA as a meta-program in Maude. Such a meta-program as a specification of CLDSA takes a concrete UDS as an actual parameter and generates the specification of the UDS-CLDSA. It is also possible to directly model check the faithful formalization of the DSR property based on the specification of the UDS-CLDSA and construct a counterexample if the property is not fulfilled.

Acknowledgement

This work was partially supported by Kakenhi 23220002, 26240008, 30272991. The authors are grateful to the anonymous reviewers who carefully read an earlier version of the paper and gave us useful comments that made it possible for us to complete the present paper.

REFERENCES

[1] Coulouris, G. F.—Dollimore, J.—Kindberg, T.—Blair, G.: Distributed Systems: Concepts and Design. 5th Edition. Addison-Wesley, 2011.

[2] Raynal, M.: Distributed Algorithms for Message-Passing Systems. Springer, 2013, doi: 10.1007/978-3-642-38123-2.

[3] Rittinghouse, J. W.—Ransome, J. F.: Cloud Computing: Implementation, Management, and Security. CRC Press, 2009.

[4] Spink, A.—Zimmer, M. (Eds.): Web Search: Multidisciplinary Perspectives. Springer, Information Science and Knowledge Management, Vol. 14, 2008, doi: 10.1007/978-3-540-75829-7.

[5] Baier, C.—Katoen, J.-P.: Principles of Model Checking (Representation and Mind Series). The MIT Press, 2008.

[6] Clarke, E. M.—Grumberg, O.—Peled, D. A.: Model Checking. MIT Press, 1999.

[7] Holzmann, G. J.: The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley, 2003.

[8] Cimatti, A.—Clarke, E.—Giunchiglia, E.—Giunchiglia, F.—Pistore, M.—Roveri, M.—Sebastiani, R.—Tacchella, A.: NuSMV 2: An OpenSource Tool for Symbolic Model Checking. In: Brinksma, E., Larsen, K. G. (Eds.): Computer Aided Verification (CAV 2002). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 2404, 2002, pp. 359–364, doi: 10.1007/3-540-45657-0_29.

[9] Lamport, L.: Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley Professional, 2002.

[10] Konnov, I.—Veith, H.—Widder J.: On the Completeness of Bounded Model Checking for Threshold-Based Distributed Algorithms: Reachability. In: Baldan, P., Gorla, D. (Eds.): CONCUR 2014 – Concurrency Theory. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 8704, 2014, pp. 125–140, doi: 10.1007/978-3-662-44584-6_10.

[11] Tsuchiya, T.—Schiper, A.: Verification of Consensus Algorithms Using Satisfiability Solving. Distributed Computing, Vol. 23, 2011, No. 5-6, pp. 341–358, doi: 10.1007/s00446-010-0123-3.

[12] Andriamiarina, M. B.—Méry, D.—Singh, N. K.: Revisiting Snapshot Algorithms by Refinement-Based Techniques. Computer Science and Information Systems, Vol. 11, 2014, No. 1, pp. 251–270, doi: 10.2298/CSIS130122007A.

[13] Chandy, K. M.—Lamport, L.: Distributed Snapshots: Determining Global States of Distributed System. ACM Transactions on Computer Systems (TOCS), Vol. 3, 1985, No. 1, pp. 63–75, doi: 10.1145/214451.214456.

[14] Spezialetti, M.—Kearns, P.: Efficient Distributed Snapshots. Proceeding of the 6th International Conference on Distributed Computing Systems (ICDCS 1986), 1986, pp. 382–388.

[15] Venkatesan, S.: Message-Optimal Incremental Snapshots. The Journal of Computer and Software Engineering, Vol. 27, 1993, pp. 211–231.

[16] Ogata, K.–Huyen, T. T. P.: Specification and Model Checking of the Chandy and Lamport Distributed Snapshot Algorithm in Rewriting Logic. In: Aoki, T., Taguchi, K. (Eds.): Formal Methods and Software Engineering (ICFEM 2012). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7635, 2012, pp. 87-102, doi: 10.1007/978-3-642-34281-3_9.

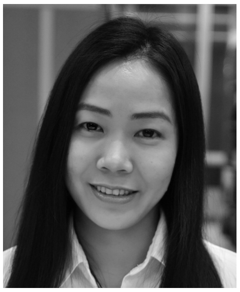
[17] Clavel, M.—Durán, F.—Eker, S.—Lincoln, P.—Martí-Oliet, N.—Meseguer, J.—Talcott, C.: All About Maude – A High-Performance Logical Framework: How to Specify, Program and Verify Systems in Rewriting Logic. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4350, 2007, doi: 10.1007/978-3-540-71999-1.

[18] Zhang, W.—Ogata, K.—Zhang, M.: A Consideration on How to Model Check Distributed Snapshot Reachability Property. IEICE Technical Report, Vol. 114, 2015, No. 416, pp. 49–54. ISSN 0913-5685.

[19] Doan, H. T. T.—Zhang, W.—Zhang, M.—Ogata, K.: Model Checking Chandy-Lamport Distributed Snapshot Algorithm Revisited. Proceeding of the 2nd International Symposium on Dependable Computing and Internet of Things (DCIT), IEEE, 2015, pp. 30–39, doi: 10.1109/DCIT.2015.13.

[20] Charron-Bost, B.—Schiper, A.: Harmful Dogmas in Fault Tolerant Distributed Computing. ACM SIGACT News, Vol. 38, 2007, No. 1, pp. 53–61, doi: 10.1145/1233481.1233496.

[21] Kshemkalyani, A. D.—Singhal, M.: Distributed Computing: Principles, Algorithms, and Systems. Cambridge University Press, 2008.



Ha Thi Thu Doan got her Ph.D. degree from the Japan Advanced Institute of Science and Technology (JAIST) in 2019. She was Lecturer at the Vietnam National University of Agriculture. She received her M.Sc. degree from the Information Science School, JAIST. Her research interests include formal methods, distributed systems, especially formal verification of distributed systems. She has been currently working on formal verification of distributed snapshot algorithms and distributed mobile robot algorithms.

Kazuhiro Ogata is Professor at the School of Information Science, Japan Advanced Institute of Science and Technology. He got his doctoral degree of engineering from the Graduate School of Science and Technology, Keio University, in 1995. Among his interesting research topics are formal methods and their application to systems, such as distributed systems.

Computing and Informatics, Vol. 38, 2019, 1039–1066, doi: 10.31577/cai_2019_5_1039

GATHERING INFORMATION ON THE WEB BY CONSISTENT ENTITY AUGMENTATION

Weijuan Sun, Ning Wang*

School of Computer and Information Technology
Beijing Jiaotong University
Beijing, 100044, P.R. China
e-mail: {15120437, nwang@}bjtu.edu.cn

Abstract. Users usually want to gather information about what they are interested in, which could be achieved by entity augmentation using a vast amount of web tables. Existing techniques assume that web tables are entity-attribute binary tables. As for tables having multiple columns to be augmented, they will be split into several entity-attribute binary relations, which would cause semantic fragmentation. Furthermore, the result table consolidated by binary relations will suffer from entity inconsistency and low precision. The objective of our research is to return a consistent result table for entity augmentation when given a set of entities and attribute names. In this paper we propose a web information gathering framework based on consistent entity augmentation. To ensure high consistency and precision of the result table we propose that answer tables for building result table should have consistent matching relationships with each other. Instead of splitting tables into pieces we regard web tables as nodes and consistent matching relationships as edges to make a consistent clique and expand it until its coverage for augmentation query reaches certain threshold γ. It is proved in this paper that a consistent result table could be built by considering tables in consistent clique to be answer tables. We tested our method on four real-life datasets, compared it with different answer table selection methods and state-of-the-art entity augmentation technique based on table fragmentation as well. The results of a comprehensive set of experiments indicate that our entity augmentation framework is more effective than the existing method in getting consistent entity augmentation results with high accuracy and reliability.

Keywords: Web table, data integration, entity augmentation, consistency

* Corresponding author

1 INTRODUCTION

In recent years, a vast amount of structured data in web pages attracts more and more attention. Google presented WEBTABLES system in 2008 [1, 2], which crawled 14.1 billion HTML tables containing 154 million relational data. Now, we can also get web tables from some commercial table search engines, such as Google Tables, Google Fusion Tables and Microsoft's Excel PowerQuery [3, 4, 5, 6]. Web tables containing relational information could be used to integrate data, one application of which is entity augmentation. Our paper focuses on entity augmentation using web tables. In other words, given a set of entities, a set of names of attributes, we regard web tables as data source to return the value of attributes for each entity.

For entity augmentation, Yakout et al. implemented InfoGather to augment entities by holistic matching [7]. They only considered binary query table with single attribute to be extended by making assumption that web tables are entity-attribute binary (EAB) relations. As for n-ary query tables, they *split* the table into several EAB relations, i.e., the subject column with each of the other columns comprise a set of EAB relations. Under this strategy, attributes in a web table are thought to be independent. Other entity augmentation systems, such as SearchJoin [8, 9, 10], use the same methods, getting result table by consolidating multiple entity-attribute binary tables.

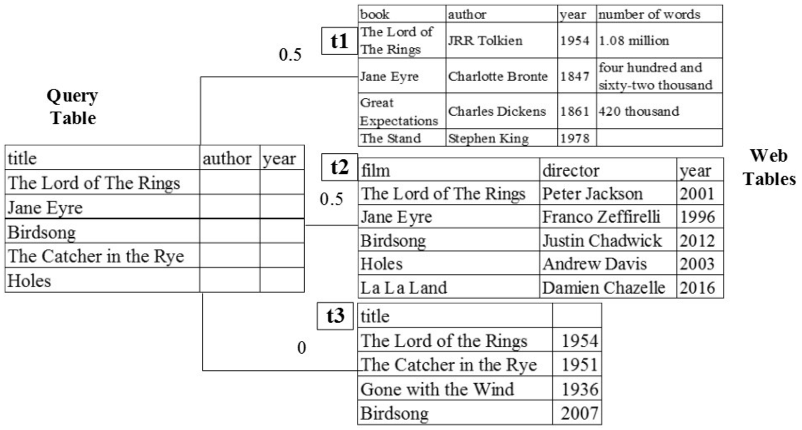
The most significant problem for current entity augmentation methods is entity inconsistency. For example, Figure 1 gives an example of entity augmentation by InfoGather, in which augmentation for only one attribute at a time could be performed. At the beginning, the second column of t_1 is selected to augment the value of column *author* in query table according to schema matching. Then, the third column of t_2 is selected to augment the value of column *year* in query table. The topic of t_1 is about book, while the topic of t_2 is about film. Consolidated by t_1 and t_2, the result table shown in Figure 2 suffers from entity inconsistency because values for *author* and *year* in the same row are not from the same entity.

The quality of the entity augmentation result could be evaluated by several aspects such as consistency, coverage and precision. By observation, we find some limitations in existing technology that leads to low coverage, low precision and low consistency.

First, existing methods usually perform entity augmentation for one attribute at a time by splitting an n-ary web table into several binary entity-attribute tables. As we discussed above, this kind of method will cause entity inconsistency and then lead to low precision.

Second, web tables usually miss header text. For example, in Figure 1, t_3 misses the label of the second attribute. When we search web tables to match the query table Q, t_3 could not be found because its attributes do not overlap Q's attributes. In this circumstance, low coverage is caused by losing some matching values.

Third, a web table is usually regarded to be matched with a query table when they both have the same entities and the same attributes. But if we only consider schema level features to find matching tables, there may be semantic conflicts be-



title	author	year
The Lord of The Rings		
Jane Eyre		
Birdsong		
The Catcher in the Rye		
Holes		

book	author	year	number of words
The Lord of The Rings	JRR Tolkien	1954	1.08 million
Jane Eyre	Charlotte Bronte	1847	four hundred and sixty-two thousand
Great Expectations	Charles Dickens	1861	420 thousand
The Stand	Stephen King	1978	

film	director	year
The Lord of The Rings	Peter Jackson	2001
Jane Eyre	Franco Zeffirelli	1996
Birdsong	Justin Chadwick	2012
Holes	Andrew Davis	2003
La La Land	Damien Chazelle	2016

title	
The Lord of the Rings	1954
The Catcher in the Rye	1951
Gone with the Wind	1936
Birdsong	2007

Figure 1. Example of entity augmentation using web tables

title	author	year
The Lord of The Rings	JRR Tolkien	2001
Jane Eyre	Charlotte Bronte	1996
Birdsong		2012
The Catcher in the Rye		
Holes		

Figure 2. The result table

tween two tables. For example, in Figure 1, t_2 is a matching table for the query table when only considering matches of entities and attribute names. However, entity inconsistency occurs because the query table is about books and t_2 is about films. This situation will also lead to low precision.

Entity inconsistency occurs when the semantics of web tables is ignored. For n-ary query table, we should regard all the attributes as a whole, instead of splitting the attributes into pieces. In the following paragraphs, we will make a distinction between an answer table and a result table. After entity augmentation, a query table Q corresponds to one result table, which contains value of attributes to be extended. In order to construct a result table, we should find a set of answer tables which provide attribute values for the final result table. Consistent entities could be obtained when answer tables match with each other either in semantics or in value, which will later be defined as consistent matching relationship in Section 2. Given a set of web tables T and a query table Q with entities, the process of entity augmentation could be thought as continuing to find answer tables from T and augmenting entities until coverage of the result table for Q reaches a specific threshold γ. If we take web tables as nodes and consistent matching relationships as edges, answer tables for a query table make up a clique because those answer tables should be matched with each other to make up consistent matching relationship.

Entity augmentation problem could be converted into another problem of building clique in a graph.

At the beginning, we find seed tables which are the most related with the query table to build initial cliques, each of which has only one node (seed table). To get a result table with a specific coverage γ, we have to expand the initial clique with more nodes (web tables) to form a clique satisfying the requirement of coverage, which is later defined as Consistent γ-Coverage Clique in Section 2. When a consistent γ-coverage clique is built successfully, the nodes in this clique are answer tables for the result table. At last, we will get a set of consistent γ-coverage cliques and corresponding answer tables, from which we choose the optimal clique and its corresponding result table.

The main contributions of this paper are:

1. We propose consistent matching relationships to solve entity inconsistency, which should be hold not only between any two answer tables but also between each answer table and the query table. We also propose answer table selection method based on consistent matching degree.
2. To our best knowledge, we are the first that propose to settle consistent entity augmentation problem by building consistent γ-coverage clique. We regard web tables having consistent matching relationships with query table as nodes and consistent matching relationships between web tables as edges, therefore we get a consistent clique. It is proved that a consistent result table could be built by considering tables in consistent clique to be answer tables.
3. We have performed extensive experiments on 4 real-life datasets of web tables. Experimental results demonstrate that our entity augmentation framework has high accuracy and reliability, meanwhile ensuring entity consistency.

This paper is structured as follows. We start in Section 2 by modeling the problem. The entity augmentation method by building consistent γ-coverage clique is described in Section 3. Experimental evaluating results are presented in Section 4. Related work is discussed in Section 5 and we conclude in Section 6.

2 PROBLEM MODELING

In recent years, entity augmentation has attracted more and more attention from researchers. Yakout et al. proposed indirect matching to augment entities [7]. Lehmberg et al. proposed Mannheim Search Join Engine to extend query table [8, 9, 10]. During entity augmentation, they only considered the entity column and the column to be augmented. The above mentioned entity augmentation techniques are all based on the assumption that columns are independent.

In case that there exist some n-ary query tables and web tables, this assumption will lead to low precision and entity inconsistency. When splitting a complete web table into several pieces, the semantics of this table is segmented, thus causing entity inconsistency in the result table and leading to low precision. In order to get consistent entity and high precision for result table, we propose that answer tables should have consistent matching relationships with each other to ensure high consistency and have consistent matching relationship with query table to ensure high precision of the result table. In order to make consistent matching relationship more understandable, we individually define concepts of semantic relevance (Definition 1) and table matching degree (Definition 2). Consistent matching relationship between two tables is made up of semantics part and value part. Semantic relevance gives the degree of how two tables are semantic related. Also, table matching degree reflects the probability that two tables consistently match in value. Before discussing the problem model, we introduce notations in Table 1.

Notation	**Description**
$Q\,(E, A)$	The query table with entity set E and attribute set A
$T = \{t_1, t_2, t_3, \ldots\}$	A set of web tables
γ	A specific coverage
AT	An answer table set for the query table
RT	The result table for the query table
$SRD(t_i, t_j)$	The semantic related degree for table t_i and t_j
$TMD(t_i, t_j)$	The table matching degree between t_i and t_j
$U\,(V, S)$	The clique U with the node set V and the edge set S
$cov\,(RT, Q)$	The coverage of result table RT to query table Q
$cov\,(U, Q)$	The coverage of clique U to query table Q
$SMS\,(Q, t)$	The semantic matching score between query table Q and web table t

Table 1. Notations

2.1 Consistent Matching Relationship

Considering two tables, we think they are semantically related if their entity sets are semantically related, because concepts of entity columns are thought to represent the tables' semantic concepts [11]. In the following paragraphs, we will first introduce how to get semantic related degree between two tables by calculating entity sets' relatedness.

We use Probase [12] to determine if two entity sets are semantically related. For each entity in one table, we calculate its relatedness to each entity in another table by using Jaccard Similarity on two entities' concept sets returned by Probase. Then, we aggregate the pairwise entity relatedness to get two tables' (t_i and t_j) semantic related degree, denoted as $SRD(t_i, t_j)$, which could be calculated using the following formula:

$$SRD(t_i, t_j) = \frac{\sum_{\forall e_i \in E_i, \forall e_j \in E_j} Jaccard(C(e_i), C(e_j))}{|E_i|\,|E_j|} \tag{1}$$

where E_i, E_j are entity sets of t_i and t_j, respectively, $C(e)$ denotes concept set of entity e, and $Jaccard\,(C\,(e_i)\,, C\,(e_j)) = \frac{|C(e_i) \bigcap C(e_j)|}{|C(e_i) \bigcup C(e_j)|}$.

Definition 1 (Semantic Relevance)**.** Given two tables t_i and t_j, we call the table t_i is semantically related with t_j, denoted as $t_i \overset{sem}{\approx} t_j$, if $SRD(t_i, t_j) \geq \theta$.

Generally, two tables are thought to match with each other in value if both tables have the same value in the same attribute for the same entities. For example, if two tables both have entity "United States", we expect them to have "Washington" in column *capital.* If there are a certain proportion of same entities with same value in column *capital* of two tables, two *capital* columns are regarded as matching columns. So, to determine if two tables are consistent matching with each other in value, we have to find their mapping columns with matching labels. Tables are matched in value if all the mapping columns are matching columns. Specially, because a query table misses attribute values, it is thought to match with a web table in value when they have same entities and same attributes. We propose the concept of *Table Matching Degree* to determine whether two tables are matched in value or not.

Definition 2 (Table Matching Degree)**.** Given two tables t_i and t_j, query table Q, C_i and C_j are respectively mapping columns for t_i and t_j. The table matching degree between t_i and t_j, denoted as $TMD(t_i, t_j)$, could be calculated using the following formula:

$$TMD(t_i, t_j) = \begin{cases} \frac{|t_i.E \bigcap t_j.E|}{\min\{|t_i.E|, |t_j.E|\}} \times \frac{|t_i.A \bigcap t_j.A|}{\min\{|t_i.A|, |t_j.A|\}}, & \text{if } t_i = Q \vee t_j = Q, \\ \frac{|\{<C_i, C_j> | C_i \approx C_j\}|}{|\{<C_i, C_j>\}|}, & \text{if } | \{< C_i, C_j >\} | \neq 0, \\ -1, & \text{otherwise,} \end{cases} \tag{2}$$

where $t.E$ denotes t's entities set, and $t.A$ denotes a set of attributes names of table t; $C_i \approx C_j$ denotes columns C_i and C_j are matching columns which meet column matching degree threshold (given in Section 3.3.2).

In Definition 2, table matching degree is -1 when there are no mapping columns between two web tables. Under this situation, whether two tables have consistent matching relationship or not is determined only by considering two tables' semantic relevance. When there are mapping columns in two tables, whether two tables have consistent matching relationship or not is determined by considering both semantic relevance and table matching degree between two tables.

Definition 3 (Consistent Matching Relationship)**.** Given two tables t_i and t_j, we call t_i has consistent matching relationship with t_j, denoted as $t_i \overset{cm}{\leftrightarrow} t_j$, iff $(t_i \overset{sem}{\approx} t_j \wedge ((TMD(t_i, t_j) \geqslant \tau \wedge t_i \neq Q \wedge t_j \neq Q) \vee TMD(t_i, t_j) = -1 \vee TMD(t_i, t_j) > 0))$.

Theorem 1. Consistent matching relationship is symmetric. Given tables t_i and t_j, if $t_i \overset{cm}{\leftrightarrow} t_j$ holds, then $t_j \overset{cm}{\leftrightarrow} t_i$ holds.

Proof. Under the condition that $t_i \overset{cm}{\leftrightarrow} t_j$, it is obvious that $t_i \overset{sem}{\approx} t_j$ and $((TMD(t_i, t_j) \geqslant \tau \wedge t_i \neq Q \wedge t_j \neq Q) \vee TMD(t_i, t_j) = -1 \vee TMD(t_i, t_j) > 0)$ hold. According to Equation (1) and Definition 1, $t_j \overset{sem}{\approx} t_i$ holds. And, according to Equation (2),

$TMD(t_i, t_j) = TMD(t_j, t_i)$ holds, which means that $((TMD(t_j, t_i) \geqslant \tau \wedge t_j \neq Q \wedge t_i \neq Q) \vee TMD(t_j, t_i) = -1 \vee TMD(t_j, t_i) > 0)$ holds. So, $t_j \overset{cm}{\leftrightarrow} t_i$ holds. That is to say, consistent matching relationship is symmetric. □

2.2 Problem Statement

For a consistent entity augmentation result, the consistent matching relationship should both exist between any two answer tables and exist between each answer table and the query table. A result table built by answer tables satisfying the above conditions is considered to be a consistent result table.

Definition 4 (Consistent Result Table)**.** Given a query table Q and a set of web tables T, RT is a result table for Q and AT is its corresponding answer tables set. RT is a consistent result table for Q if and only if:

1. Each table in AT has consistent matching relationship with the query table Q.
2. Tables in AT have consistent matching relationships with each other.

Problem Statement. Given query table $Q(E, A)$ and a set of web tables T, where $Q.E$ denotes query table's entities, and $Q.A$ denotes a set of names of attributes which are to be augmented. *Consistent Entity Augmentation* is to find a group of answer tables AT ($AT \subseteq T$) for building a consistent result table RT whose coverage to query table reaches the specific threshold γ.

If we take web tables having consistent matching relationships with query table as nodes and consistent matching relationships as edges, then a clique is the complete subgraph in which tables as nodes consistently match with each other. Tables in a clique come to be answer tables when their corresponding result table's coverage to query table is larger than the specific threshold. So, entity augmentation problem could be converted into building consistent γ-coverage clique (Definition 7) problem.

Definition 5 (Result Coverage)**.** Given a query table Q, a clique U composed of answer tables for Q and the corresponding result table RT, the coverage of result table RT or clique U to Q, denoted as $cov(RT, Q)$ or $cov(U, Q)$, could be calculated using the following formula:

$$cov(RT, Q) = cov(U, Q) = \frac{\#\text{augCells}(RT)}{\#Cells(Q)} \tag{3}$$

where $\#\text{augCells}(RT)$ and $\#Cells(Q)$ denote the number of cells augmented by the clique U, and the number of cells which should be augmented in the query table, respectively.

Definition 6 (Consistent Clique)**.** Given a query table Q and a set of candidate tables CT, a clique $U(V, S)$ is a consistent clique for Q when:

- V is a subset of CT, and each table in V has consistent matching relationship with Q;
- S is a set of consistent matching relationships between table pairs in V, $\forall(t_i, t_j) \in S$, $t_i \overset{cm}{\leftrightarrow} t_j$ holds;

where CT is selected from source web tables T, having at least one same entity with the query table.

Definition 7 (Consistent γ-Coverage Clique)**.** Given a query table Q, a consistent clique $U(V,S)$ for Q, and a coverage threshold γ, a clique U is a consistent γ-coverage clique for Q when $cov(U,Q) \geq \gamma$.

Theorem 2. Given a query table Q, a set of web tables T, a consistent result table RT for Q with γ coverage could be built using V as answer tables if and only if existing a consistent γ-coverage clique $U(V,S)$ for Q.

Proof. Firstly, we prove necessity. When a consistent result table RT for Q with γ coverage exists, we could regard its answer tables as nodes set V and consistent matching relationships between tables as edges set S, then get a complete graph $U(V,S)$. According to Definition 4, consistent matching relationships in RT exist both between any two answer tables and between each answer table and the query table, which makes the complete graph $U(V,S)$ to be a consistent clique for Q. Furthermore, under the condition that coverage of result table RT to Q reaches γ, it is obvious that $U(V,S)$ is a consistent γ-coverage clique for Q.

Secondly, we will prove sufficiency. If existing a consistent γ-coverage clique $U(V,S)$, according to Definition 7, every table in V has consistent matching relationship with Q. For clique U, $t_i \overset{cm}{\leftrightarrow} t_j$ holds for any two tables t_i, t_j in V. Using V as answer tables, a consistent result table RT could be built for Q. Furthermore, when $cov(U,Q) \geq \gamma$, $cov(RT,Q) \geq \gamma$ holds. □

Theorem 2 is given to validate that we could get a consistent result table for query table Q with γ coverage by building Consistent γ-Coverage Clique. Based on this theorem, the entity augmentation problem could be converted into the *Consistent γ-Coverage Clique Problem* as follows.

The Consistent γ-Coverage Clique Problem. Given a query table Q and a set of web tables T, the consistent γ-coverage clique problem is to build a set of cliques whose coverage to query table Q are larger than γ.

3 ENTITY AUGMENTATION BY BUILDING CONSISTENT γ-COVERAGE CLIQUE

3.1 Solution Overview

Given the query table is missing some attributes' values, as one application of data integration, the aim of entity augmentation is to return a result table containing query table's missing data. Existing entity augmentation techniques assume web tables are EAB relations [7, 8, 9, 10]. When an n-ary query table has multiple columns to be extended, the result table consolidated by binary tables will suffer from entity inconsistency problem. The goal of our paper is to build consistent results for n-ary entity augmentation queries.

Figure 3 gives the framework for consistent entity augmentation. At first, we find candidates from source tables with aid of the index $EI(Q)$. Given a query table Q and a set of web tables T, index $EI(Q)$ will return tables having at least one same entity with the query table. In order to get consistent entity for result table, we propose that answer tables should have consistent matching relationships with each other to ensure high consistency and precision of the result table. Based on graph theory, consistent entity augmentation query problem could be converted into consistent γ-coverage clique building problem. We also prove that tables in consistent γ-coverage clique are answer tables for entity augmentation query in Theorem 2.

To build γ-coverage cliques, we first find seed cliques as initials using *semantic matching score*. For each seed clique, we try to expand it with tables and get a corresponding consistent γ-coverage clique. For a seed clique not satisfying the requirement of coverage, we have to find other web tables called clique tables to improve its coverage. In order to get clique tables, we calculate table potential for each candidate table, which is made up of its consistent matching degree with the query table and its consistent matching degree with each table in clique. Obviously, a web table is selected to be a clique table, if it has consistent matching relationship not only with query table but also with each table in existing clique. In other words, the higher is the table potential, the greater is the probability of being a clique table. Based on this intuition, we select one with the highest potential as a new clique table, and add edges between this new addition and each original clique table. After that, we continue to expand the clique by adding tables with high potential until its coverage reaches γ. For each seed table, we will get a consistent γ-coverage clique and corresponding answer tables. To get final consistent result table, we select the optimal clique by measuring several indicators such as consistency support degree, diversity of source and coverage. At last, tables in optimal clique are regarded as answer tables which provide attribute values for the final consistent result table.

3.2 Finding Seed Cliques Based on Semantic Relevance

In our solution, the first step is to find seed cliques, which is the base for building consistent γ-coverage cliques. In most cases, a query table contains less information, so seed cliques are introduced to provide as much information as possible for entity augmentation. As shown in Figure 1, "The Lord of The Rings" is not only a best-selling novel, but it is also a popular movie. The information provided by the query table is not enough to determine entity's concept. If we provide more information, such as *publishing time*, it will be helpful for identifying concept of the entity. So, it is very important to find seed cliques with tables "matching with" query table.

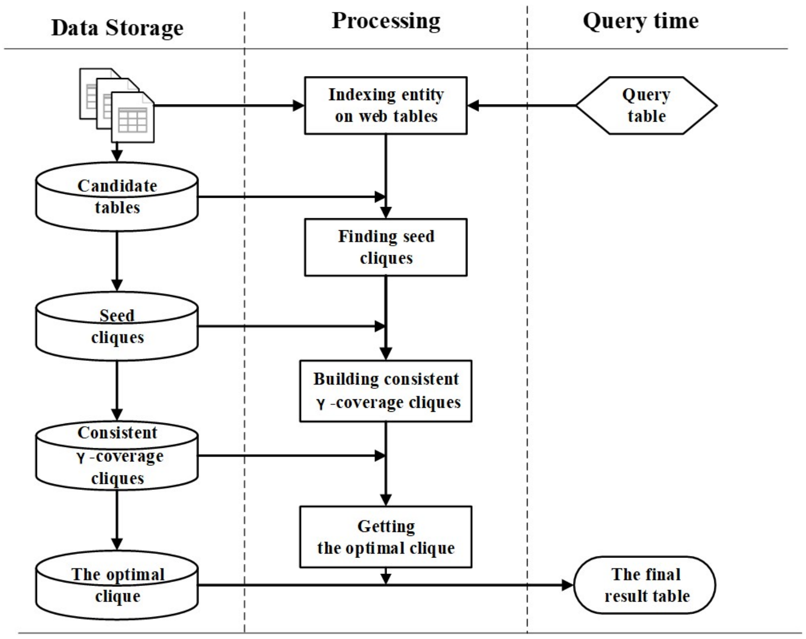


Figure 3. A framework for consistent entity augmentation

At the beginning, we can get a graph made up of isolated tables without consistent matching relationships among them. At that time, each node in the graph could be regarded as an initial clique, which would be expanded by other nodes (web tables) to form a consistent γ-coverage clique. In order to improve accuracy and reduce time cost, we should select seed cliques having more consistent matching degree with the query table as initial cliques. Because each initial clique only has one node, the problem of finding seed cliques could be converted to the problem of finding seed tables.

To find seed tables, schema matching techniques are usually used, which is generally based on schema level features (e.g. attribute names) and instance level features (e.g. attribute values). Previous works only consider schema information and select seed tables when they have the same entities and same attribute names with the query table [7]. Using the existing method, for example in Figure 1, we can get seed tables t_1 and t_2 because their matching scores are the same. Obviously, t_2 is a false seed table whose entities are films, while entities of query table are books. During entity augmentation, error will be amplified by using t_2 as the seed table.

This problem is due to only considering schema information. In fact, based on schema level features, we can also consider semantic relevance between a candidate table and the query table. And, according to *Problem Statement* described in Section 2.2, every answer table should have a consistent matching relationship with the query table, so are seed tables. As for selection of seed tables, we hope that seed tables have a higher consistent matching degree than other tables, based on that they all have consistent matching relationships with a query table. To measure consistent matching degree between a web table and a query table, we need to calculate *semantic matching score.*

Definition 8 (Semantic Matching Score)**.** Given the query table $Q(E, A)$ and a web table $t(K, B)$, the semantic matching score between Q and t, denoted as $SMS(Q, t)$, could be calculated using the following formula:

$$SMS(Q,t) = \phi(SRD(Q,t), \theta) * \phi(TMD(Q,t), 0) \tag{4}$$

where $\phi(p, \theta) = [\![p > \theta]\!]_{-\infty}^{p}$, which denotes $\phi(p, \theta) = p$ when $p > \theta$, otherwise $\phi(p, \theta) = -\infty$. $SRD(Q, t)$ denotes semantic related degree between query table and web table. When $SRD(Q, t) < \theta$, we think the table t and the query table are not semantically related.

Given the query table and candidate tables CT, for each table in CT, we can get its semantic matching score with query table Q. According to semantic matching scores, we can get top-k seed tables, which makes the initial cliques taking shape.

3.3 Building Consistent γ-Coverage Cliques

For any seed clique, we need to expand it by adding web tables (denoted as clique tables) if its coverage is smaller than γ. Based on seed cliques, we can build consistent γ-coverage clique by dynamically adding clique tables. According to Definition 6, a clique is considered as a consistent clique when its nodes have a consistent matching relationships with the query table and its edges are consistent matching relationships. Through Definition 8, every seed clique meets the requirement of being a consistent clique. The way to find a clique table is considering its consistent matching degree both with the query table and with each table in the current clique. A node potential reflects consistent matching degree between the clique table and the query table, while an edge potential reflects consistent matching degree between two tables in the clique. The potential of a candidate table is the sum of its node potential and all edge potentials between itself and each table in the clique.

3.3.1 Node Potential

Node potential is proposed to measure the probability of a candidate to become a clique table, which mainly considers the coverage of the clique after adding the web table into it. In order to represent the contribution of a web table for increasing clique's coverage, we propose the concept of *Supplementary Coverage.*

Definition 9 (Supplementary Coverage)**.** Given a query table $Q(E, A)$, candidate tables CT and a consistent clique $U(V, S)$ for Q whose coverage is smaller than γ.

The supplementary coverage of a web table t ($t \in CT - V$) about the clique U for the query table Q, denoted as $SC(t, U, Q)$, is the incremental coverage of clique U improved by table t.

$$SC(t, U, Q) = cov(U \cup \{t\}, Q) - cov(U, Q). \tag{5}$$

Node Potential. Given a query table $Q(E, A)$, candidate tables CT and a consistent clique $U(V, S)$ for Q whose coverage is smaller than γ. Node potential $\varphi_{node}(t, U, Q)$ for table t ($t \in CT - V$) is calculated as follows:

$$\varphi_{node}(t, U, Q) = \phi(SRD(Q, t), \theta)) * SC(t, U, Q). \tag{6}$$

3.3.2 Edge Potential

For web tables missing labels, Equation (6) could not give accurate node potential for those tables. Using Figure 1 as an example, the supplementary coverage of t_3 is 0 due to missing a column label in t_3. Actually, with label *year* in the second column, t_3's supplementary coverage should be 0.2 based on seed table t_1. We can settle this problem by transferring the label *year* from the third column of t_1 to the second column of t_3 by finding matching columns.

Two columns will be matching columns when their column matching degree is larger than a specific threshold. For two columns C_i and C_j in tables t_i and t_j, respectively, their column matching degree, denoted as $CM(C_i, C_j)$, reflects the similarity degree of two columns. Column matching degree is the ratio of the same entities with the same values in two columns. For column matching degree of two columns, we mainly consider three situations:

1. both columns' elements are character values;
2. both columns' elements are years;
3. both columns' elements are numeric values.

For character values, two values are regarded as the same when their EditDistance is larger than a specific threshold. For two years, two values are regarded as the same when they equal to each other. Specially, for numeric values, two values are regarded as same when they meet a specific unit conversion.

Definition 10 (Matching Columns)**.** Given two tables t_i and t_j, C_i and C_j are columns for t_i and t_j, respectively, they are regarded as matching columns, denoted as $C_i \approx C_j$, if $CM(C_i, C_j) > \sigma$ holds.

In order to calculate node potential for a web table with missing column labels, we try to transfer labels of columns which are query table's mapping columns to their matching columns.

After transferring labels, we can get edge potential over the table pair when two tables have a consistent matching relationship with each other. Edge potential denotes consistent matching degree between two tables, considering two tables' consistent matching degree both in semantics and in value.

Edge Potential. Given candidate tables CT and a consistent clique $U(V,S)$ for a query table Q whose coverage is smaller than γ. Edge potential $\varphi_{edge}(t_i,t_j)$ for edge between table t_i in $CT-V$ and table t_j in V could be calculated using following formula:

$$\varphi_{edge}(t_i,t_j)=\begin{cases}\phi(SRD(t_i,t_j),\theta), & \text{if } TMD(t_i,t_j)=-1,\\ \frac{\phi(SRD(t_i.t_j),\theta)+\phi(TMD(t_i.t_j),\tau)}{2}, & \text{otherwise.}\end{cases} \tag{7}$$

3.3.3 Getting Clique Tables

The goal of this stage is continuously to find clique tables from the set of candidate tables CT, to make the coverage of clique approach γ. So, to find clique table for clique $U(V,S)$, we regard the sum of node potential and edge potential as table potential, denoted as $\varphi_{table}(t_i)$.

$$\varphi_{table}(t_i)=\varphi_{node}(t_i,U,Q)+\sum_{\forall t_j\in V}\varphi_{edge}(t_i,t_j) \tag{8}$$

where $t_i\in CT-V$.

According to Equation (8), we can get the clique table t_U for the clique $U(V,S)$ using the following formula:

$$t_U=\underset{\forall t_i\in CT-V}{\arg\max}\,\varphi_{table}(t_i). \tag{9}$$

In order to build consistent γ-coverage clique, we select seed tables first according to semantic matching score. Then, for each seed table whose coverage is smaller than γ, we separately calculate table potentials for candidates and select one with the maximal value as clique table. Each time when new clique table is putting in, the clique is expanded and its coverage is improved. Above progress is repeated until clique's coverage reaches γ.

Algorithm 1 gives an algorithm for finding clique table. The input of this algorithm is the query table Q, candidate tables set CT and the clique $U(V,S)$ whose coverage is smaller than γ. At first, labels from matching columns are transferred to tables with missing labels (line 8). Then, table potential is calculated for each candidate table (line 9). At last, the clique table t with the maximal table potential is chosen (line 11–12).

Algorithm 2 introduces the procedure of iteratively building consistent γ-coverage cliques based on seed cliques. The input is the query table Q, candidate tables

```
Algorithm 1: findCTables($Q(E,A), CT, U(V,S)$)
  Input: $Q(E,A)$: the query table;
         $CT$: candidate table set;
         $U(V,S)$: a clique whose coverage is smaller than $\gamma$;
  Output: $t_U$: clique table for $U(V,S)$
1  $map \leftarrow \emptyset$;
2  for each table $t_i$ in $CT - V$ do
3      $sum \leftarrow 0$;
4      for each table $t_j$ in $V$ do
5          for each column $C_j$ in $t_j$' column set $\Gamma$ used to augment $Q$ do
6              for each column $C_i$ in $t_i$ do
7                  if $CM(t_i, t_j) > \sigma$ and $A(C_i) \neq A(C_j)$ then
8                      $A(C_i) \leftarrow A(C_j)$;
9          $sum \leftarrow sum + \varphi_{edge}(t_i, t_j)$;
10     add $\langle t_i, sum + \varphi_{node}(t_i, U, Q)\rangle$ to $map$;
11 $map = HashMap(map)$;
12 return $top - 1$ table in $map$;
```

set CT and the number k of seed cliques. At the beginning, the set of initial cliques is empty (line 1). Then, we get seed cliques (line 2). For each seed clique, the function $expand\_Clique(Q, CT, U_i(V,S), \varepsilon)$ is executed iteratively until the clique's coverage reaches γ (line 4). In practice, sometimes when the result table returned could not satisfy the requirement of coverage γ, we would return the clique whose coverage is the closest to γ instead. Under this situation, a clique U is returned when its coverage converges (line 13–14).

3.4 Getting Consistent Result Table Based on Optimal Clique

After getting a set of consistent γ-coverage cliques, we can obtain the corresponding answer tables for the query. At this stage, we should select an optimal clique, whose nodes are answer tables for building the final consistent result table. We consider the following indicators for selecting the optimal clique:

Consistency Support Degree: We measure consistency support degree using the average value of table potentials in clique $U(V,S)$. A higher value means the result table could keep higher consistency with the query table.

Diversity of Source: This indicator reflects diversity of answer tables (i.e., the diversity of tables in clique $U(V,S)$). We measure diversity of source simply by the number of answer tables. Generally, the more diverse the data source is, the less consistent the result table will be. In fact, fewer answer tables are helpful for maintaining the entity consistency.

Algorithm 2: *build_CC Cliques*(Q, CT, k)

```
Input: $Q(E, A)$: the query table;
       $CT$: candidate table set;
       $k$: the number of seed cliques;
Output: $\mathbb{C}$: a set of consistent $\gamma$-coverage cliques
1  $\mathbb{C} \leftarrow \varnothing$;
2  get the seed cliques $U_1, U_2, \cdots, U_k$;
3  for each seed clique $U_i(V, S)$ do
4      $U_i(V, S) \leftarrow expand\_Clique(Q, CT, U_i(V, S), \varepsilon)$;
5      add $U_i(V, S)$ in $\mathbb{C}$;
6  return $\mathbb{C}$;
7
8  Function $expand\_Clique(Q, CT, U_i(V, S), \varepsilon)$
9  if $cov(U_i, Q) \geq \gamma$ then
10     return $U_i(V, S)$;
11 else
12     $t \leftarrow findCTables(Q(E, A), CT, U_i(V, S))$;
13     if $SC(t, U_i, Q) \leq \varepsilon$ then
14         return $U_i(V, S)$;
15     else
16         add $t$ in $V$; for each $t_i$ in $V$ do
17             add $\langle t, t_i \rangle$ in $S$;
18         $expand\_Clique(Q, CT, U_i(V, S), \varepsilon)$;
```

Coverage: Even though we set the coverage threshold, the coverage of result tables returned by different cliques will be different. Apparently, we prefer clique with a higher coverage.

We select optimal clique to build the final consistent result table by using the following formula.

$$U_{final} = \arg\max_{U \in \mathbb{C}} \frac{\varphi(U) * cov(U, Q)}{|V|^2} \quad (10)$$

where $\varphi(U) = SMS(Q, t_{seed}) + \sum_{t_i \in U.V - \{t_{seed}\}} \varphi_{table}(t_i)$ and $t_{seed} \in U.V$, $\mathbb{C}$ is a set of consistent γ-coverage cliques returned by Algorithm 2.

After getting the optimal clique, we could build a consistent result table using answer tables in this clique. Algorithm *get_RT* describes the procedure with answer tables AT, query table Q and the similarity threshold α as input. In Algorithm 3, array *flag* is a sign array used to record whether a cell in the result table has been filled or not (line 1). To avoid entity inconsistency, the fill-in progress is based on tables. By sorting answer tables according to their *semantic matching*

Algorithm 3: $get\_RT(AT, Q(E,A), \alpha)$

```
Input: $Q(E,A)$: the query table;
       $AT$: answer tables set;
       $\alpha$: the similarity threshold;
Output: $RT$: the final result table
1 $flag[] \leftarrow 0, RT \leftarrow Q$;
2 while $AT! = \varnothing$ do
3     $t \leftarrow \arg\max_{t \in AT} SMS(Q,t)$;
4     for each $rt.cell \in RT$ do
5         if $flag[rt.cell] = 0 \&\& \arg\max_{t.cell \in t} sim(rt.cell, t.cell) \geq \alpha$ then
6             $t.cell_{max} \leftarrow \arg\max_{t.cell \in t} sim(rt.cell, t.cell)$;
7             $flag[rt.cell] \leftarrow 1$;
8             $rt.cell(v) \leftarrow t.cell_{max}(v)$;
9     $AT \leftarrow AT - \{t\}$;
```

score, we select all available information from tables in turn to fill the result table (line 2–9).

4 EXPERIMENTS

We are the first who have proposed to settle consistent entity augmentation problem by building a consistent γ-coverage clique. We conduct a set of experiments to evaluate the effectiveness of our method (denoted as EACC) for getting high quality of entity augmentation results by selecting answer tables based on consistent matching degree. In the following paragraphs, we first evaluate the effectiveness of our answer table selection method based on consistent matching degree, then evaluate the performance of our consistent entity augmentation system based on building consistent γ-coverage clique.

Four real-life datasets (Books, Country, Company and Song) from WDC Web Table Corpora (`http://webdatacommons.org/webtables/`) and four query tables in Table 2 are used in our experiments. We compiled the complete ground truth for query tables by manually identifying and extracting the desired information from Wikipedia [13].

Dataset Name	**Entity**	**Attributes to be Augmented**	**Number of Entities**
Books	Book name	Author, Published date	101
Country	Country name	Capital, Area	26
Company	Company name	Headquarter, Industry	48
Song	Song name	Artist, Time	47

Table 2. Features of query tables

All methods were written in Java and all tests were conducted on a PC with a 2.93 GHz Intel CPU and 8 GB RAM, running Windows 7.

4.1 Effectiveness of Our Answer Table Selection Method Based on Consistent Matching Degree

The effectiveness of our answer table selection method based on consistent matching degrees is evaluated by comparing other three methods as follows:

1. EACC: It is our entity augmentation method based on consistent matching relationship and consistent matching degree, which finds answer tables by building consistent γ-coverage clique.
2. EATSP: It uses topic sensitive pagerank (TSP) [14] to select answer tables and the corresponding result tables. TSP score is computed for each table as its matching degree with the query table. EATSP implements entity augmentation queries according to TSP score of each table.
3. EAWOS: It selects answer tables depending on Table Matching Degree calculated only by value similarity between tables, which could be regarded as EACC method without semantics.
4. DMA: Different from above three methods, DMA is a kind of direct matching approach only considering web tables which match directly with the query table. It selects a web table as an answer table according to consistent matching degree between this web table and the query table.

As we know, InfoGather is a typical entity augmentation system which selects answer tables by TSP scores. For n-ary queries, however, InfoGather splits tables into several EAB relations, while our method EACC regards all attributes in a table as a whole. Instead of splitting attributes into pieces, EATSP regards all the attributes as a whole but selects answer tables by TSP scores. Compared with EACC, EAWOS considers only value matching but ignoring semantic matching between tables. EACC, EASTP and EAWOS are all approaches which consider indirectly matching tables in addition to the directly matching ones. The direct match approach and indirect match approach are described in [7]. DMA is a naive method that attempts to directly match the query table with the web tables. The comparison of four entity augmentation methods are listed in Table 3.

Augmentation Method	**Matching Mode**	**Matching Score**
EACC	Indirect & Direct	Consistent matching degree
EASTP	Indirect & Direct	TSP
EAWOS	Indirect & Direct	Table matching degree
DMA	Direct	Consistent matching degree

Table 3. Features of four methods

We extracted 16 GB web tables from WDC Web Table Corpora as data source and deleted some off-grade web tables – such as tables with confusing information, more empty values, and ambiguous subject columns. There are 3 000 000 web tables in the data source in which the maximum number of rows and columns are 454 and 42, and the minimum number of rows and columns are 3 and 2. The average number of rows and columns in web tables are 13 and 5. We run four methods respectively to augment four query tables in Table 2 and compare their performance. Under different coverage threshold (varying from 0.1 to 0.8), their coverage, precision, consistency and reliability are compared. The evaluation metrics are as follows.

$$cov = \frac{|values\_found|}{\#Cells(Q)}, \tag{11}$$

$$pre = \frac{|values\_found \bigcap values\_truth|}{|values\_found|}, \tag{12}$$

$$con = \begin{cases} 1, & \text{if } |AT| = 1, \\ 1 - |\log_{10}[\![avgSim(t_1,t_2) \geq 0.1]\!]_{0.1}^{avgSim(t_1,t_2)}|, & \text{otherwise}, \end{cases} \tag{13}$$

where $avgSim(t_1,t_2) = avg \sum_{t_1,t_2 \in AT} \frac{SRD(t_1,t_2)+TMD(t_1,t_2)}{2}$.

Coverage is the ratio of filled values to values to be filled. Precision is the fraction of correctly filled values that have been really filled. Values extracted from Wikipedia are regarded as truth values. Consistency is measured by the average value of similarity between any two answer tables. Obviously, answer tables having high consistency will receive a high similarity. *Semantic relevance* and *table matching degree* between two tables are considered for evaluating their similarity. Inspired by F-measure, reliability is defined as harmonic mean of coverage, consistency and precision.

$$relia = \frac{3 * cov * pre * con}{cov * pre + pre * con + cov * con}. \tag{14}$$

4.1.1 Evaluation for Quality of Result Tables

We have implemented four methods for entity augmentation: EACC, EATSP, EAWOS and DMA. Figures 4, 5, 6 show respectively coverage, consistency, and precision for four query tables. We varied results by considering different coverage threshold from 0.1 to 0.8.

From Figures 4, 5, 6, we have the following observations:

1. With the increase of γ, the coverage results of four methods are increasing. In most cases, the coverage of EATSP and DMA is higher than that of others under different coverage thresholds. Answer tables in EATSP and DMA should only match with the query table, while answer tables in EACC and EAWOS should

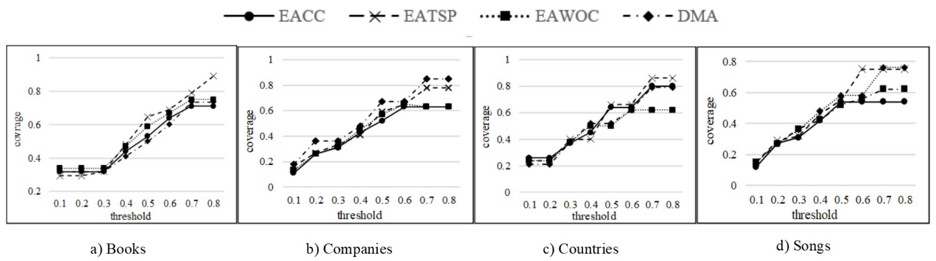


Figure 4. Coverage values on four datasets

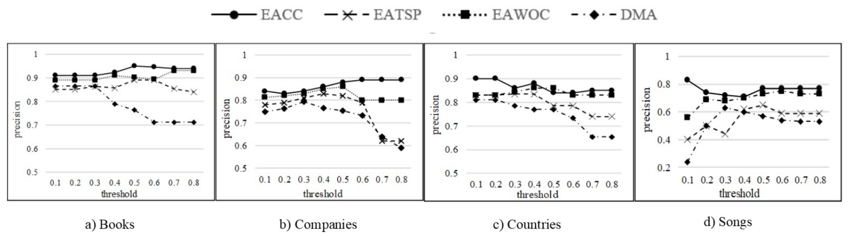


Figure 5. Precision values on four datasets

meet consistent matching relationships with each other, which causes less answer tables to be left in the clique. On most datasets, EATSP's coverage is higher than that of DMA, because EATSP considers indirectly matching tables in addition to the directly matching ones, and DMA only considers directly matching tables.

2. On four datasets, the precision of EACC and EAWOS is obviously higher than that of others, because two methods all get answer tables based on matching relationships between web tables. By comparison, EACC is better than EA-WOS. EACC considers consistent matching relationships either in semantics or in value, but EAWOS only considers table matching relationships in value, which greatly reduces precision. EATSP gets answer tables based on topic sensitive

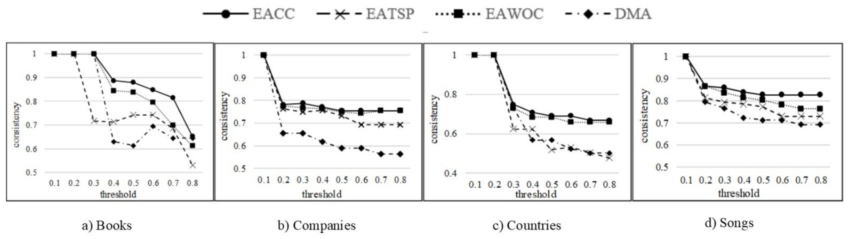


Figure 6. Consistency values on four datasets

pagerank algorithm (TSP), considering mainly schema level features among tables and text features on web page. Due to ambiguity of entity and redundancy of web page information, EATSP gets lower precision. In Figure 5, EACC based on consistent matching degree significantly outperforms EATSP, EAWOS and DMA, which confirms that precision results could be improved by considering consistent matching relationship.

3. For we evaluate the consistency mainly based on the average similarity value between answer tables, the consistency value reaches the highest when there is only one answer table. With the increase of coverage threshold, most consistency results of four algorithms on four datasets decrease due to the increasing number of answer tables. In most cases, the consistency results of our EACC are greater than EAWOS, EATSP and DMA, because EACC selects answer tables not only considering consistent matching degree between each candidate table and the query table but also consistent matching degree among answer tables.

Experimental results show that EACC algorithm based on consistent matching degree significantly outperforms EATSP, EAWOS and DMA in precision and consistency even though the coverage of EACC is a little lower than that of three other algorithms in some cases.

4.1.2 Evaluation for Reliability of Result Tables

Based on coverage, precision and consistency, we can get the reliability of result tables under different coverage thresholds using Equation (14). Figure 7 shows the experimental results.

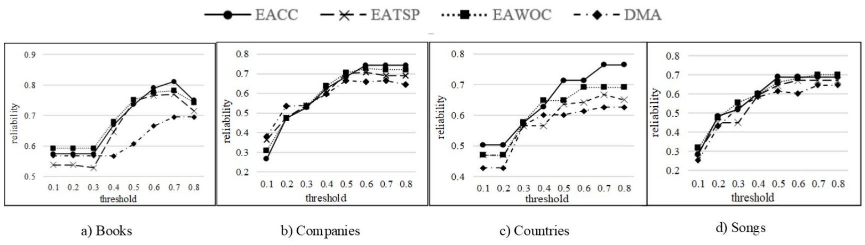


Figure 7. Reliability values on four datasets

With the increase of coverage thresholds, the general trends of reliability results for four algorithms are increasing. For reliability which is the harmonic mean of precision, coverage and consistency, EACC has the highest reliability. As the threshold grows, the reliabilities of four algorithms reach highest (0.81 for EACC, 0.76 for EATSP, 0.78 for EAWOS and 0.695 for DMA).

In summary, even though the coverage of EACC is a little lower than that of three other algorithms in some cases, EACC has highest precision, consistency and reliability, which will be helpful for returning consistent entity augmentation results. According to above comparison experiments, we come to the conclusion that our answer table selection method based on consistent matching degree either in semantics or in value is helpful for returning effective and consistent result table for entity augmentation queries.

4.1.3 Evaluation for Runtime Performance

We evaluate the runtime performance of the given four methods. We give the runtime performance with the increase of coverage thresholds in Figure 8.

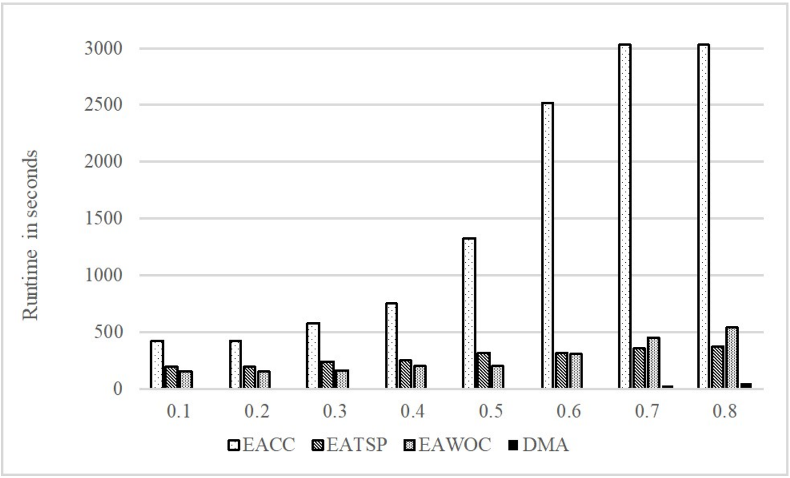


Figure 8. Runtime values on different coverage thresholds

In Figure 8, with the increase of coverage threshold, EACC's runtime has the most obvious growth trend and relatively high runtime. That is because EACC costs a lot of time in searching semantics and creating cliques meeting the requirement of coverage. EAWOS's runtime is obviously lower than EACC, because selection of answer tables in EAWOS does not consider semantics. In Figure 8, we can find that the runtime of EATSP varies little under different coverage thresholds, for it just calculates matching degree between each candidate and the query table, no matter what coverage threshold is. DMA is a naive method that attempts to directly match the user query table with the web tables, which takes the least time.

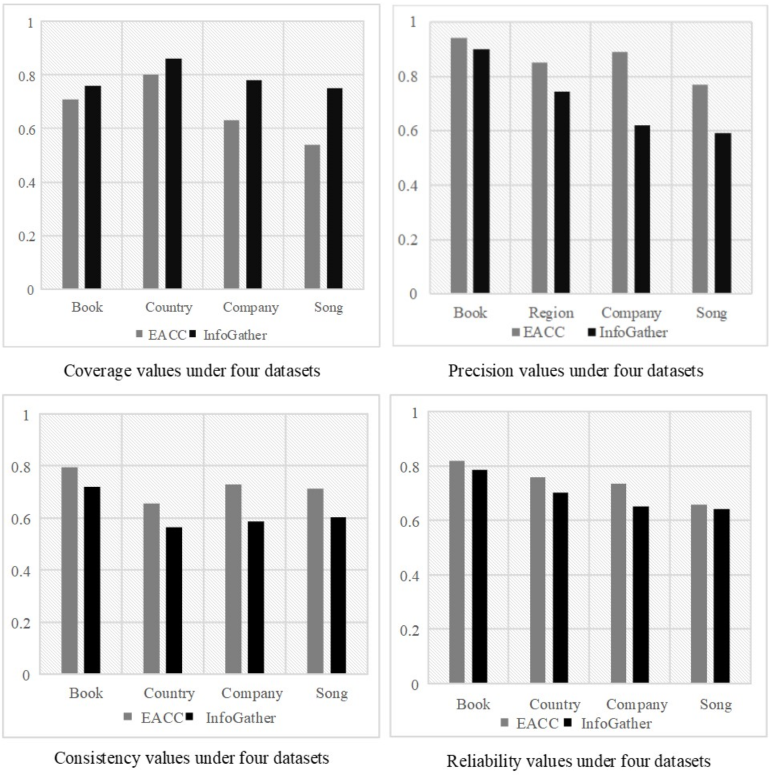
Although EACC takes much time for searching semantics and creating cliques meeting the requirement of coverage, it has prominent performance in precision, consistency and reliability for entity augmentation. The experiments demonstrate that consistent matching relationship proposed by this paper is much helpful for settling entity inconsistency problem caused by existing methods. Using parallel algorithms to improve the efficiency of consistent entity augmentation method is our future work.

4.2 Performance of Consistent Entity Augmentation

Web tables are assumed as EAB relations in existing entity augmentation techniques. InfoGather is such a typical entity augmentation system proposed by Mohamed Yakout, that is based on graphical models and topic sensitive pagerank algorithm. To compare our algorithm EACC with InfoGather, we use coverages of result tables returned from InfoGather as EACC's coverage thresholds.

Different from InfoGather which answers n-ary queries by splitting attributes into pieces, our EACC method augments entities by building consistent γ-coverage cliques based on consistent matching degree.

We do experiments to compare the performance of EEAC and InfoGather, including their coverage, precision, consistency and reliability. The experimental results are given in Figure 9.



Coverage values under four datasets

Precision values under four datasets

Consistency values under four datasets

Reliability values under four datasets

Figure 9. Entity augmentation comparison between EACC and InfoGather

From Figure 9, we have the following observations:

1. The coverages of InfoGather on four datasets are higher than that of EACC. EACC requires answer tables to have consistent matching relationships not only with each other but also with the query table, which certainly decreases the number of answer tables.
2. The average precisions (over all four datasets) of EACC and InfoGather are 0.86 and 0.71, respectively, demonstrating that EACC significantly outperforms InfoGather in precision. The consistency of EACC is also higher than that of InfoGather because InfoGather composes the result table from many data sources on a per-entity basis. And, InfoGather augments entities by splitting tables into several EAB relations, which can easily lead to entity inconsistency. Our EACC can get higher precision and higher consistency by considering semantic relevance and consistent matching relationships between tables.
3. Consequently, for reliability which is the harmonic mean of precision, coverage and consistency, our EACC method performs better than InfoGather. In Figure 9, the result set on Song dataset has the largest coverage difference between EACC and InfoGather. However, it has the minimum reliability difference between EACC and InfoGather. By observation, we find Song is the largest among four datasets. In fact, the larger the dataset is, the more answer tables there will be. Due to the restriction of consistent matching relationship, EACC will get less answer tables than InfoGather. Meanwhile, EACC will get higher precision and consistency. So, as the harmonic mean, the reliability difference between two methods on Song dataset will be smaller.

In summary, our method EACC performs much better than InfoGather in precision and consistency. Experimental results demonstrate that our entity augmentation framework has high accuracy and reliability, meanwhile ensuring also entity consistency.

5 RELATED WORK

At present, a large body of research work is about web search and data integration. Entity augmentation refers to extend attribute content based on entity or other known information, which helps people to obtain information they are interested in by web tables.

Web tables are important data source for gathering information by entity augmentation. Compared to other data sources such as knowledge base and crowdsourcing with human intelligence, web tables are more open and comprehensive. To our best knowledge, WEBTABLES system presented by Cafarella et al. is the first work on using the wealth of web tables [1, 2]. The authors extracted a large scale corpus of web tables and proposed several applications for such a corpus. They introduced AcsDB which enables several novel applications such as schema auto-complete, attribute synonym finding and join-graph traversal. According to user-supplied keywords, WEBTABLES returns a list of web tables based on relevance ranking, and users can browse and filter useful information in the tables. WEBTABLES system gives us a chance to know the huge potential of web tables. Based on WEBTABLES system, Balakrishnan et al. display web table data as rich snippets in search engine results [15].

To integrate structured data, some researchers proposed to collect information from various data sources into a single table according to a set of keywords. Cafarella et al. proposed Octopus to integrate structured data [16]. Octopus used web search API to retrieve a ranked list of matching tables and integrated tables into a single table according to user interactions. MultiJoin algorithm was proposed to implement EXTEND operator in Octopus, which could find matching web tables for each queried entity independently and then cluster the web tables found. Pimplikar et al. presented a search engine which returned a multi-column result table in response to a keyword query without any known entities [17]. To achieve the table query, they mainly want to know if a web table is relevant to the query table, and if so, label each column of the web table with the query column to which it maps. The authors converted this task into a graphical model which took mappings of all candidate table columns into consideration. In the recent years, human intelligence has been introduced in the information processing and management. Park et al. implemented CrowdFill for collecting structured data from the crowd [18, 19], in which the interaction between workers and requesters is realized by Amazon Mechanical Turk. A partially-filled table is shown in CrowdFill to all participating workers, and workers contribute by filling in empty cells, as well as upvoting and downvoting data entered by other workers to return requesters the results collection.

There are also some studies for augmenting missing information in tables. Gupta et al. proposed an end to end system named WWT [20], which consolidated a table from a few example rows by harnessing the huge corpus of information-rich but unstructured lists on the web. InfoGather [7] is a system to augment binary tables. It identifies not only tables directly matched with the query table but also tables indirectly matched with the query table, what greatly improves the coverage of entity augmentation. InfoGather+ [21] is an improvement on InfoGather, which answers entity augmentation queries accurately for numeric and time varying attributes. It assigns labels for time and units of measurements of tables, and propagates labels among two connected nodes in semantic graph. However, InfoGather+ also argues that tables are entity-attribute binary relations, which hence leads to semantic fragmentation. Besides, Lehmberg et al. designed Search Join, a search engine achieving the search, join and composition of tables [8, 9, 10]. Eberius et al. use consistent set covering to solve the diversity of query results, and return users top-k result tables [22]. The authors proposed to process these queries in top-k fashion, in which they produced multiple minimal consistent solutions from which the user can choose to resolve the uncertainty of the data sources and methods used. However, all above systems consider the entity column as the only basis to augment another attribute column, ignoring the association between attributes and correlation between tuples.

For entity augmentation, a major challenge is the fact that many web tables have missing or non-informative column labels. To solve the problem, Braunschweig et al. proposed to identify and extract column specific information from the context of web tables [23]. They proposed a heuristic approach to extract column specific context information for relational tables on the Web. And, He et al. focus on automatic discovery of attribute synonyms [24]. They mainly consider attribute synonymy from query click logs and web table attribute name co-occurrences. The authors formalized the problem as an optimization problem on a graph, with the attribute names being the vertices and the positive and negative evidences from query logs and web table schemas as weighted edges. They developed a linear programming based algorithm to solve the problem. The method of discovering attribute synonyms is beneficial to improve the accuracy and coverage of entity augmentation. For some web tables with small size, Lehmberg et al. proposed that stitching web tables into a larger one could improve matching quality [25]. Above works are orthogonal to the problem of entity augmentation in this paper.

6 CONCLUSIONS

In this paper, we present the EACC framework to achieve consistent entity augmentation queries by using web tables as data source. In order to solve the entity inconsistency problem in existing technology, we propose the consistent matching relationship which should be hold between answer tables, and convert the problem of entity augmentation into the problem of building consistent γ-coverage cliques. Experimental results demonstrate that our entity augmentation framework has high accuracy and reliability, meanwhile ensuring the entity consistency. There are some future works, such as using parallel algorithms to improve the efficiency of big data processing, using the crowdsourcing platform to make full use of human intelligence to verify or correct result tables, and so on.

Acknowledgment

This work is supported by the National Key R & D Program of China (2018YFC080-9800), and by the National Natural Science Foundation of China (61370060, 616730-48).

REFERENCES

[1] Cafarella, M. J.—Halevy, A.—Wang, D. Z.—Wu, E.—Zhang, Y.: WebTables: Exploring the Power of Tables on the Web. Proceedings of the VLDB Endowment, Vol. 1, 2008, No. 1, pp. 538–549, doi: 10.14778/1453856.1453916.

[2] Cafarella, M. J.—Halevy, A.—Wang, D. Z.—Wu, E.—Zhang, Y.: Uncovering the Relational Web. Proceedings of the 11th International Workshop on Web and Databases (WebDB 2008), Vancouver, Canada, 2008.

[3] Google Tables. Available at: `https://research.google.com/tables`.

[4] Gonzalez, H.—Halevy, A. Y.—Jensen, C. S.—Langen, A.—Madhavan, J.—Shapley, R.—Shen, W.—Goldberg-Kidon, J.: Google Fusion Tables: Web-Centered Data Management and Collaboration. Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD '10), 2010, pp. 1061–1066, doi: 10.1145/1807167.1807286.

[5] Gonzalez, H.—Halevy, A.—Jensen, C. S.—Madhavan, J.—Shapley, R.—Shen, W.: Google Fusion Tables: Data Management, Integration and Collaboration in the Cloud. Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC '10), 2010, pp. 175–180, doi: 10.1145/1807128.1807158.

[6] Microsoft's Excel PowerQuery. Available at: `https://office.microsoft.com/powerbi`.

[7] Yakout, M.—Ganjam, K.—Chakrabarti, K.—Chaudhuri, S.: InfoGather: Entity Augmentation and Attribute Discovery by Holistic Matching with Web Tables. Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD '12), 2012, pp. 97–108, doi: 10.1145/2213836.2213848.

[8] Bizer, C.: Search Joins with the Web. Christian Science Monitor, 2014.

[9] Lehmberg, O.—Ritze, D.—Ristoski, P.—Eckert, K.—Paulheim, H.—Bizer, C.: Extending Tables with Data from over a Million Websites. Semantic Web Challenge, International Semantic Web Conference, Riva del Garda, Italy, 2014.

[10] Lehmberg, O.—Ritze, D.—Ristoski, P.—Meusel, R.—Paulheim, H.—Bizer, C.: The Mannheim Search Join Engine. Journal of Web Semantics, Vol. 35, 2015, Part 3, pp. 159–166, doi: 10.1016/j.websem.2015.05.001.

[11] Sarma, A. D.—Fang, L.—Gupta, N.—Halevy, A.—Lee, H.—Wu, F.—Xin, R.—Yu, C.: Finding Related Tables. Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD '12), 2012, pp. 817–828, doi: 10.1145/2213836.2213962.

[12] Wu, W.—Li, H.—Wang, H.—Zhu, K. Q.: Probase: A Probabilistic Taxonomy for Text Understanding. Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD '12), 2012, pp. 481–492, doi: 10.1145/2213836.2213891.

[13] Wikipedia. Available at: `http://www.wikipedia.org`.

[14] Haveliwala, T. H.: Topic-Sensitive PageRank. Proceedings of the 11th International Conference on World Wide Web (WWW '02), 2002, pp. 517–526, doi: 10.1145/511446.511513.

[15] Balakrishnan, S.—Halevy, A.—Harb, B.—Lee, H. et al.: Applying WebTables in Practice. Proceedings of the 7th Biennial Conference on Innovative Data Systems Research (CIDR '15), Asilomar, California, USA, 2015.

[16] Cafarella, M. J.—Halevy, A.—Khoussainova, N.: Data Integration for the Relational Web. Proceedings of the VLDB Endowment, Vol. 2, 2009, No. 1, pp. 1090–1101, doi: 10.14778/1687627.1687750.

[17] Pimplikar, R.—Sarawagi, S.: Answering Table Queries on the Web Using Column Keywords. Proceedings of the VLDB Endowment, Vol. 5, 2012, No. 10, pp. 908–919, doi: 10.14778/2336664.2336665.

[18] Park, H.—Widom, J.: CrowdFill: Collecting Structured Data from the Crowd. Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14), 2014, pp. 577–588, doi: 10.1145/2588555.2610503.

[19] Park, H.—Widom, J.: CrowdFill: A System for Collecting Structured Data from the Crowd. Proceedings of the 23rd International Conference on World Wide Web (WWW '14 Companion), 2014, pp. 87–90, doi: 10.1145/2567948.2577029.

[20] Gupta, R.—Sarawagi, S.: Answering Table Augmentation Queries from Unstructured Lists on the Web. Proceedings of the VLDB Endowment, Vol. 2, 2009, No. 1, pp. 289–300, doi: 10.14778/1687627.1687661.

[21] Zhang, M.—Chakrabarti, K.: InfoGather+: Semantic Matching and Annotation of Numeric and Time-Varying Attributes in Web Tables. Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13), 2013, pp. 145–156, doi: 10.1145/2463676.2465276.

[22] Eberius, J.—Thiele, M.—Braunschweig, K.—Lehner, W.: Top-k Entity Augmentation Using Consistent Set Covering. Proceedings of the 27th International Conference on Scientific and Statistical Database Management (SSDBM '15), 2015, Art. No. 8, pp. 1–12, doi: 10.1145/2791347.2791353.

[23] Braunschweig, K.—Thiele, M.—Eberius, J.—Lehner, W.: Column-Specific Context Extraction for Web Tables. Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC '15), 2015, pp. 1072–1077, doi: 10.1145/2695664.2695794.

[24] He, Y.—Chakrabarti, K.—Cheng, T.: Automatic Discovery of Attribute Synonyms Using Query Logs and Table Corpora. Proceedings of WWW Conference, 2016, pp. 1429–1439, doi: 10.1145/2872427.2874816.

[25] Lehmberg, O.—Bizer, C.: Stitching Web Tables for Improving Matching Quality. Proceedings of the VLDB Endowment, Vol. 10, 2017, No. 11, pp. 1502–1513, doi: 10.14778/3137628.3137657.

Weijuan Sun received her Master degree in computer science from the Beijing Jiaotong University in 2018 and currently works in Baidu Netcom Science and Technology (Beijing) Co., Ltd. Her main research areas include web data integration and big data management.

Ning Wang received her Ph.D. degree in computer science in 1998 from the Southeast University in Nanjing, China. She is currently serving as Professor in the School of Computer and Information Technology, Beijing Jiaotong University, China. Her research interests include web data integration, big data management, data quality and crowdsourcing.

Computing and Informatics, Vol. 38, 2019, 1067–1090, doi: 10.31577/cai_2019_5_1067

NEW APPROACH TO EDGE DETECTION ON DIFFERENT LEVEL OF WAVELET DECOMPOSITION

Vladimir Maksimović, Branimir Jakšić, Mile Petrović
Petar Spalević, Stefan Panić

Faculty of Technical Science
University of Priština
Kneza Miloša 7
38220 Kosovska Mitrovica, Serbia
e-mail: {vladimir.maksimovic, branimir.jaksic, mile.petrovic, petar.spalevic}@pr.ac.rs, stefanpnc@tpu.ru

Abstract. This paper proposes a new approach to edge detection on the images over which the wavelet decomposition was done to the third level and consisting of different levels of detail (small, medium and high level of detail). Images from the BSD (Berkeley Segmentation Dataset) database with the corresponding ground truth were used. Daubechies wavelet was used from second to tenth order. Gradient and Laplacian operators were used for edge detection. The proposed approach is applied in systems where information is processed in real time, where fast image processing is required and in systems where high compression ratio is used. That is, it can find practical application in many systems, especially in television systems where the level of details in the image changes. The new approach consists in the fact that when wavelet transform is applied, an edge detection is performed over the level 1 image to create a filter. The filter will record only those pixels that can be potential edges. The image is passed through a median filter that filters only the recorded pixels and 8 neighbors of pixel. After that, the edge detection with one of the operators is applied onto the filtered image. F measure, FoM (Figure of Merit) and PR (Performance Ratio) were used as an objective measure. Based on the obtained results, the application of the proposed approach achieves significant improvements and these improvements are very good depending on the number of details in the image and the compression ratio. These results and improvements can be used to improve the quality of edge detection in many systems where compressed images are processed, that is, where work with images with a high compression ratio is required.

Keywords: Edge detection, wavelet decomposition, compression, F measure, figure of merit, performance ratio

1 INTRODUCTION

In recent years, multimedia systems have recorded tremendous growth and progress, which means that image resolutions got higher increasing the complexity of the system and the complexity of the image. It is necessary to achieve as much compression as possible so these images can be streamed, stored and processed more easily. Today's trends require more technology to be involved, so in television systems we have a virtual reality (VR), augmented reality (AR) and a combination of them. Since it is necessary to have lower bitrate and achieve a high compression ratio and to process an image in VR or AR environment, it is necessary to perform certain operations over images such as segmentation, edge detection, etc. [1].

Edge detection is one of the fundamental processes in image processing. This also means the segmentation of the image where the segmentation of the desired object is performed by detecting the edge. Edge detection is based on the fact that there are sudden changes in the gray intensity between the objects. Edge detection significantly reduces the image analysis process by using less data and at the same time storing all the necessary information. Many edge detection techniques have been tried, but gradient and Laplacian methods have proved to be the best [2, 3]. Gradient methods for edge detection find the maximum and minimum gradient of the image intensity, all in the first derivative of the image. Laplacian methods are based on finding zero crossing in the second derivative of the image [4]. The gradient of the image can be calculated as [4, 5]:

$$\nabla f\left(x, y\right) = \frac{\partial f}{\partial x} i + \frac{\partial f}{\partial y} j \tag{1}$$

where $f\left(x, y\right)$ represents the image at the location $\left(x, y\right)$ where x and y are the coordinates of the row and column. The gradient $\nabla f\left(x, y\right)$ contains the information about gray change. The gradient of $\nabla f\left(x, y\right)$ can be calculated [4, 5]:

$$e\left(x, y\right) = \sqrt{f_x^2 + f_y^2} \tag{2}$$

where $e\left(x, y\right)$ can be used as an edge detector and can also be defined as the sum of the absolute values of the partial derivative f_x and f_y [4, 5]:

$$e\left(x, y\right) = \left|f_x\left(x, y\right)\right| + \left|f_y\left(x, y\right)\right|. \tag{3}$$

Based on these theoretical principles, the gradient and Laplacian methods are proposed. The gradient methods include Sobel, Prewitt, Robert, while the Laplacian include LoG (Laplacian of Gaussian). The classic use of the Canny operator is a gradient method, but in some parts, it also contains elements of the Laplace approach [6, 7, 8].

The analysis of the frequency domain using Fourier transform is very useful for signal analysis because the frequency domain is very important for the consideration of the nature of the signal and the influence of the noise over it. The disadvantage of this technique is the loss of time domain information. When viewing the Fourier transform in a frequency domain, it is difficult to say when a certain event occurred, or when some frequencies occurred. To overcome this problem, only a small part ("window") of the signal is analyzed at a time, and this window slides over the signal applying the Fourier transform on it. In this way, part of the frequency information is lost in order to get an information about the time when a certain frequency has occurred. Wavelet transformation is used to solve this problem. It allows the use of different window sizes for different frequencies. The basic difference between the wavelet transformation and the short-time Fourier transform is that the window length changes at the wavelet transform and in this way the frequency and time resolution can be changed. The basic idea of each wavelet transform is to present an arbitrary function $x(t)$ as a superposition of a wavelet set or basic functions. The basic functions are derived from a prototype called mother wavelet, by scaling or translating this function. The wavelet transformation of the $x(t)$ signal is given in Equation (4) [9, 10]:

$$\psi\left(\tau, s\right) = \frac{1}{s}\int_{-\infty}^{\infty} x\left(t\right) * \psi^{*}\left(\frac{t-\tau}{s}\right) \mathrm{d}t \tag{4}$$

where τ is a translation, and s is a scaling, while ψ is a "mother" wavelet, and ψ^{*} is conjugally complex of ψ. When processing images in a spatial domain, the operation is discretized. Discrete wavelet transformation (DWT) is defined as [9, 10]:

$$DWT_{x}^{\psi}\left(\tau, s\right) = \frac{1}{\sqrt{|s|}}\int_{-\infty}^{\infty} x\left(t\right) * \psi^{*}\left(\frac{t-\tau}{s}\right) \mathrm{d}t. \tag{5}$$

The discrete wavelet transformation (DWT) can be presented as a matrix $\psi\left(c\right)$ and DWT coefficients can be obtained by taking an internal product between the signal and the wavelet matrix [9, 10]:

$$DWT_{x}^{\psi}\left[n, s\right] = \frac{1}{\sqrt{|s|}}\sum_{n} x\left[n\right]\psi_{n,s}^{*}\left[n\right]. \tag{6}$$

Special family of wavelet functions has been developed for DWT. They are generally divided into orthogonal or biorthogonal and are characterized by a high-pass and low-pass filters [11]. Daubechies wavelet belongs to a family of orthogonal functions and the main feature is the possibility of the maximum number of vanishing moments for a predefined supported length. Types of Daubechies (db) wavelets that are most commonly used in practical applications are dbN, where N represents the order as well as the number of vanishing moments in the supported interval from 0 to 20. By increasing the order of the Daubechies wavelet, better characteristics are obtained, but the complexity of the implementation, the price of the system and errors in the calculations rise. In practical applications, orders 2 to 10 are most commonly used [10, 12, 13].

A very important advantage of the wavelet transform is that it uses a multiresolution analysis that allows analysis of different signal frequencies at different frequency resolutions. For high-frequency parts, shorter windows are used, which ensures good time domain resolution, while longer parts of the lower frequencies have longer windows, thus giving good information about the frequencies. If the signal is passed through a set of two filters, low-pass and high-pass, its frequency content will be split into two equal-width ranges. The output from these filters contains half the frequency of the original signal and the same number of samples as the original signal. By decimating, or by passing the input signal through the low-pass filter, the number of samples is halved so that the time resolution is also halved while the frequency resolution increases. The high-pass filter transmits high-frequency content, i.e. signal details. The low-pass signal transmits low-frequency content, or signal approximation. This approximation signal can be further fed through two filters and the process can be repeated until the desired decomposition level is reached. The complete information on the original signal is contained in the last approximation signal and all the detail signals [14, 15].

Higher compression ratio disrupts the quality of the image, resulting in a large loss of information, and hence makes it difficult to process them, for example, to do an edge detection or facial recognition [16, 17]. The decomposition significantly degrades the image quality, but besides this degradation there are various types of noise that further impair the image quality. To solve this problem or to control it, the various types of filters have been developed [18, 19].

In this paper we used the characteristics of wavelet transform in order to image compression, or as a method on which some algorithms for image compression is based, such as JPEG2000 and SPIHT. Images from the BSD database are compressed to the third decomposition level which resulted in a high degree of compression.

2 SYSTEM MODEL AND PROPOSED APPROACH

In this paper, the BSD (Berkeley Segmentation Dataset) image database was used for analysis, with its corresponding ground truth images [20]. Table 1 lists the selected images from the BSD database meeting the complexity criteria [16], that is, each image consists of a different level of detail: small, medium, and high level of detail. The number of details was calculated by making DCT and DWT on the high-frequency components (details), which are divided into four quadrants, along both directions (x and y). After that, the mean absolute value of the amplitude of the components belonging to the quadrants is calculated [16]: DCT in quadrant 1 (dctd); DCT in quadrants 2 and 3 (dctm); DWT in quadrant 1 (dwtd); DWT in quadrants 2 and 3 (dwtm).

	Images	**dctd**	**dctm**	**dwtd**	**dwtm**
Criterion L	#135 069	1.544	2.517	0.181	0.354
Criterion M	#35 010	3.838	6.197	1.199	2.048
Criterion H	#8 143	7.868	15.241	3.181	6.336

Table 1. Criteria

Figure 1 gives a flow diagram of the proposed approach. In the proposed approach, firstly, the BSD and the ground truth images are read on which the DWT is applied to the third decomposition level using the Daubechies wavelet (optionally from 2 to 20). Since grayscale images are needed, a conversion is made, and variables are created for filters and new images. In order to assign input values to a filter, it is necessary to perform edge detection on level 1 images. Values stored in the filter are information about location of pixels pointing at the potential edge. The compressed image on which the detection is performed is a binary image and passing through the loop, each pixel is examined. If it is equal to the 1, that is, there is an edge, the median filter is used on the pixel and 8 neighbors of pixel. In other words, only those pixels that are relevant for the edge detection are passed through the filter. After that, the detection is applied and the results are compared, as shown in Figure 1. The algorithm is applied to all edge detection operators (Canny, LoG, Sobel, Prewitt, Robert).

The algorithm consists of the following steps:

Step 1 (Read image): Read the original image. If it is a color image, converts it to a grayscale image.

Step 2 (DWT): DWT is applied to the third decomposition level onto a read image, whereby as a result, three images are obtained: image from level 1, image from level 2 and image from level 3. The next steps in the algorithm are applied separately for an image from each level.

Step 3 (Edge detection): On level 1 image, an edge detection is used by selecting one of detectors (Canny, LoG, Sobel, Prewitt, Roberts). The image with detected edges serves to create a filter that will contain only those pixel coordinates where the edges are located.

Step 4 (Filtering image): Every pixel in the image is analyzed and if it is equal to 1, the filter records its coordinates.

Step 5 (Filtering 8-neighbors of pixel): If the condition in step 4 is fulfilled, filtering is done by filtering 8 neighbors of pixel relative to the current pixel using the median filter (Figure 2). So, only those image pixels that can be edges are filtered. The neighbors of pixel are extracted using the following formula:

If current pixel $P(i,j)$ is edge then use following neighbor pixels:

$$f(P_{i,j}) = f(P_{i-1,j-1}), f(P_{i-1,j}), f(P_{i-1,j+1}), f(P_{i,j-1}), f(P_{i,j}), f(P_{i,j+1}),\\ f(P_{i+1,j-1}), f(P_{i+1,j}), f(P_{i+1,j+1}) \tag{7}$$

and use median filter only on those pixels.

Step 6 (Edge detection on filtered image): Edge detection is applied on a filtered image.

For filtering level 2 and level 3 images, coordinates from level 1 image are used.

The proposed algorithm has a quadratic complexity and can be represented as:

$$O(row, col) = 8(row - 2)(col - 2) + 1. \tag{8}$$

In Figure 3, the complexity of the algorithm is given, depending on the number of pixels in the row (row) and the number of pixel in columns (cols).

In order to accurately and precisely present how well the edge detection is done, it is necessary to calculate Precision, Specificity, Sensitivity and Accuracy or F measure [21]. F measure (F1 score) is a harmonic mean of precision and recall and it combines precision and recall according to the formula [21]:

$$F = \frac{2 * Precision * Recall}{Precision + Recall} \times 100. \tag{9}$$

F is within the limits of $0 \leq F \leq 1$, ideally, F is equal to 1. In the results, F is multiplied by 100 and represents a percentage value.

Figure of Merit (FoM) was proposed by Pratt [22] and is a measure for estimating the accuracy of detected edges. In other words, it represents the deviation of the actual (calculated) point of the edge from the ideal point of the edge, and is defined as:

$$FoM = \frac{1}{\max[I_d, I_i]} \sum_{k=1}^{I_d} \frac{1}{1 + \delta e^2(k)} \tag{10}$$

where I_d is the number of points on the detected edge, and I_i is the number of points on the ideal edge, $e(k)$ represents the distance between the detected edge and the ideal edge, and δ is scaling constant and is usually 1/9. FoM is within $0 \leq FoM \leq 1$. In this case, FoM is multiplied by 100 and represents a percentage value. The higher the FoM is, the better the detected edge is [22].

As an objective measure of the edge detection credibility, Performance Ratio (PR) was used too. The PR is ideally equal to infinity. The PR is calculated as the ratio of the true edges to false ones [23, 24]:

$$PR = \frac{\text{True Edge(Edge pixels identified as Edges)}}{\text{False Edges(Non Edge pixels identified as Edges)+(Edge pixels identified as Non Edge pixels)}} \times 100. \qquad (11)$$

3 RESULTS

3.1 F Measure

Tables 2, 3 and 4 show the F values before and after the application of the proposed approach on image with a small, medium and high number of details, respectively, over which db (from 2nd to 10th order) wavelet transform to the third decomposition level was applied. These tables show the values obtained for the five operators.

From Table 2 it can be seen that based on the obtained results, in the first decomposition level, using the proposed approach, improvements were achieved with the Canny operator in almost all cases, except for db4. A similar situation occurs when a LoG operator is used, with the exception that improvements have not been achieved in the case of db6. Using the proposed approach and image with low details, significant improvements have been achieved at the second level, where all operators record improvements in F values, except in the case of db8 with Prewitt and Sobel operators. In the third level, only Canny and LoG record improvements in F values using the proposed approach.

In the image with a medium number of details, very small improvements have been achieved in some operators at the first level, or the values are generally similar, without major deviations, as can be seen in Table 3. The situation is different with the second and third decomposition levels, where improvements are achieved by all operators, with the difference that improvements are higher in the third level.

In the case where the image consists of a high number of details, using the proposed approach, a similar or slightly better F value is obtained. At the second level, improvements are generally achieved with gradient operators, while at the third level, using the proposed approach, better values are obtained for all operators.

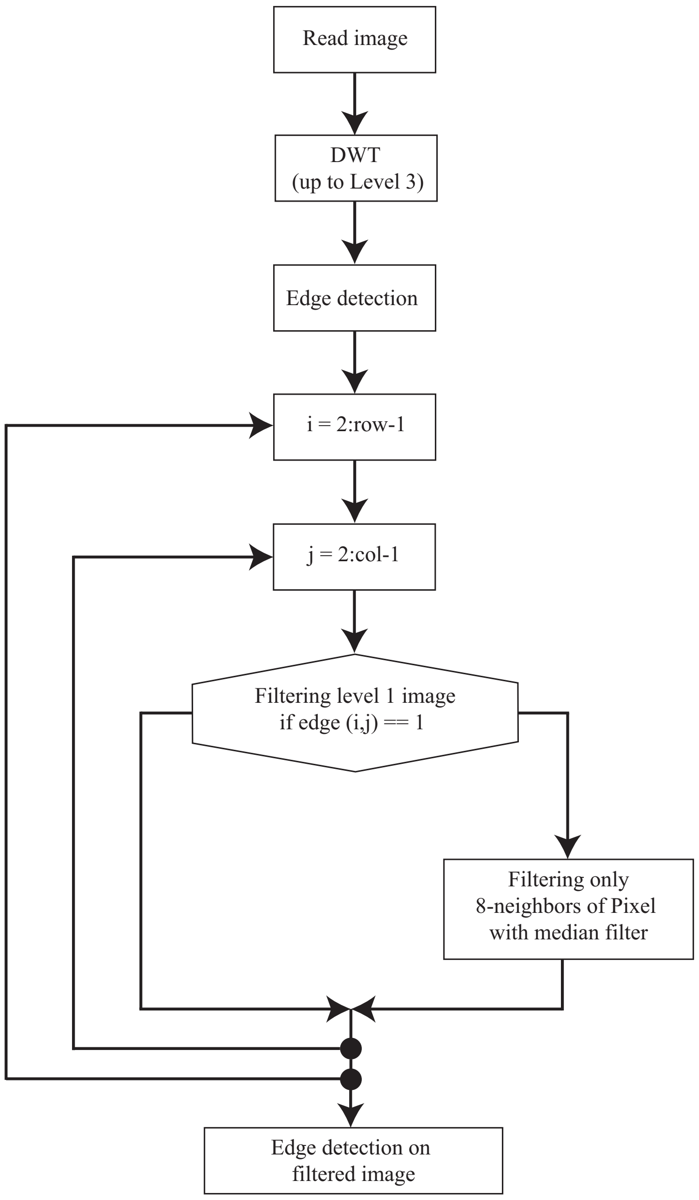


Figure 1. Proposed approach

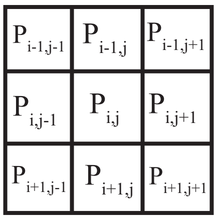


Figure 2. Extraction neighbors of pixel

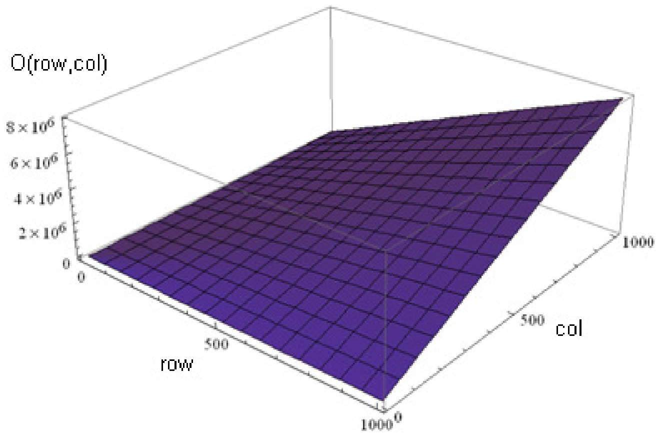


Figure 3. Complexity of proposed algorithm

Wavelet		**db2**		**db4**		**db6**		**db8**		**db10**	
	Operator	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**
Level1	**Canny**	0.365	**0.378**	0.379	0.364	0.368	**0.380**	0.363	**0.365**	0.371	**0.375**
	LoG	0.311	**0.314**	0.313	**0.317**	0.311	0.299	0.300	**0.303**	0.311	**0.315**
	Prewitt	0.362	0.342	0.355	0.346	0.366	**0.364**	0.322	**0.345**	0.380	0.362
	Sobel	0.365	0.336	0.356	0.349	0.371	0.363	0.325	**0.346**	0.377	0.362
	Roberts	0.536	**0.563**	0.503	**0.579**	0.519	**0.576**	0.551	**0.566**	0.514	**0.582**
Level2	**Canny**	0.278	**0.305**	0.246	**0.269**	0.203	**0.225**	0.189	**0.194**	0.194	**0.211**
	LoG	0.256	**0.274**	0.235	**0.256**	0.191	**0.211**	0.205	**0.207**	0.186	**0.202**
	Prewitt	0.269	**0.302**	0.285	**0.339**	0.285	**0.321**	0.338	0.301	0.279	**0.351**
	Sobel	0.264	**0.306**	0.284	**0.320**	0.281	**0.314**	0.333	0.303	0.278	**0.333**
	Roberts	0.299	**0.416**	0.350	**0.453**	0.397	**0.456**	0.448	**0.486**	0.433	**0.495**
Level3	**Canny**	0.108	**0.160**	0.113	**0.169**	0.096	**0.170**	0.100	**0.158**	0.106	**0.172**
	LoG	0.128	**0.173**	0.148	**0.190**	0.132	**0.187**	0.148	**0.174**	0.144	**0.176**
	Prewitt	0.147	0.203	0.189	0.241	0.184	0.247	0.180	0.251	0.194	0.256
	Sobel	0.146	0.201	0.187	0.229	0.182	0.235	0.179	0.240	0.195	0.244
	Roberts	0.183	0.276	0.208	0.255	0.218	0.297	0.256	0.335	0.268	0.348

Table 2. F values for an image with a low number of details (LD)

Wavelet		**db2**		**db4**		**db6**		**db8**		**db10**	
	Operator	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**
Level1	**Canny**	0.318	0.316	0.322	0.318	0.321	0.317	0.321	0.319	0.321	0.320
	LoG	0.234	0.234	0.235	**0.236**	0.235	**0.236**	0.236	0.236	0.237	0.236
	Prewitt	0.299	0.222	0.219	**0.220**	0.225	0.225	0.218	**0.219**	0.226	0.226
	Sobel	0.229	0.223	0.220	0.219	0.226	0.226	0.217	**0.219**	0.227	0.225
	Roberts	0.250	0.249	0.267	**0.268**	0.268	0.258	0.274	0.267	0.282	0.270
Level2	**Canny**	0.295	**0.302**	0.299	**0.312**	0.297	**0.312**	0.298	**0.312**	0.300	**0.309**
	LoG	0.213	0.211	0.207	**0.213**	0.211	**0.213**	0.209	**0.214**	0.207	**0.212**
	Prewitt	0.205	**0.209**	0.170	**0.182**	0.160	**0.184**	0.158	**0.182**	0.164	**0.185**
	Sobel	0.205	**0.209**	0.169	**0.180**	0.161	**0.182**	0.158	**0.178**	0.163	**0.183**
	Roberts	0.153	**0.154**	0.157	**1.196**	0.158	**0.183**	0.155	**0.177**	0.162	**0.179**
Level3	**Canny**	0.186	**0.238**	0.199	**0.246**	0.215	**0.252**	0.203	**0.250**	0.204	**0.250**
	LoG	0.134	**0.153**	0.111	**0.135**	0.088	**0.111**	0.081	**0.104**	0.076	**0.100**
	Prewitt	0.118	**0.138**	0.092	**0.118**	0.078	**0.115**	0.072	**0.110**	0.075	**0.110**
	Sobel	0.118	**0.135**	0.092	**0.114**	0.078	**0.109**	0.072	**0.104**	0.075	**0.105**
	Roberts	0.078	**0.081**	0.065	**0.088**	0.042	**0.054**	0.034	**0.046**	0.035	**0.047**

Table 3. F values for an image with a medium number of details (MD)

Wavelet		**db2**		**db4**		**db6**		**db8**		**db10**	
	Operator	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**
Level1	**Canny**	0.222	0.219	0.219	**0.220**	0.220	0.219	0.219	**0.220**	0.220	**0.221**
	LoG	0.192	0.192	0.190	0.189	0.192	0.192	0.192	**0.193**	0.192	0.191
	Prewitt	0.171	0.167	0.164	**0.166**	0.178	0.175	0.165	**0.166**	0.172	0.169
	Sobel	0.170	0.165	0.164	**0.165**	0.177	0.174	0.163	**0.165**	0.170	0.168
	Roberts	0.161	0.152	0.166	0.163	0.163	0.157	0.170	0.162	0.164	0.157
Level2	**Canny**	0207	**0.207**	0.216	**0.224**	0.205	**0.215**	0.215	**0.220**	0.214	**0.220**
	LoG	0.177	0.176	0.182	0.183	0.172	**0.176**	0.174	**0.175**	0.179	**0.182**
	Prewitt	0.159	**0.166**	0.139	**0.145**	0.127	**0.143**	0.135	**0.141**	0.133	**0.150**
	Sobel	0.157	**0.166**	0.139	**0.144**	0.127	**0.142**	0.133	**0.140**	0.133	**0.147**
	Roberts	0.123	0.123	0.129	**0.167**	0.125	**0.143**	0.135	**0.149**	0.142	0.159
Level3	**Canny**	0.145	**0.184**	0.154	**0.175**	0.154	**0.197**	0.161	**0.190**	0.162	**0.190**
	LoG	0.121	**0.148**	0.104	**0.133**	0.094	**0.114**	0.084	**0.108**	0.077	**0.102**
	Prewitt	0.110	**0.136**	0.077	**0.102**	0.078	**0.114**	0.073	**0.103**	0.077	**0.111**
	Sobel	0.109	**0.133**	0.077	**0.098**	0.078	**0.112**	0.073	**0.100**	0.078	**0.110**
	Roberts	0.087	**0.089**	0.064	**0.088**	0.067	**0.084**	0.064	**0.079**	0.070	**0.087**

Table 4. F values for an image with a high number of details (HD)

3.2 Figure of Merit

Tables 5, 6 and 7 show the FoM values before and after the application of the proposed approach on image with a small, medium and high number of details, respectively, over which db (from 2^{nd} to 10^{th} order) wavelet transform to the third decomposition level was applied. These tables show the values obtained for the five operators.

From Table 5 it can be seen that in the first decomposition level, the best values are obtained for the Canny operator, where in almost all cases it gives better values, except in the case of db4. By comparing Table 5 with Table 2, it can be seen that using FoM objective measurements, improvements have been made in the same or similar cases. By increasing the compression, that is, in Levels 2 and Levels 3, using the proposed approach, better FoM values were obtained, and greater improvements were achieved, especially in the third level.

In the case of an image with a medium number of details, in the first level, Canny gave the best results, in other words, better values were achieved. In the second level, in the case of db2, improvements were achieved only with the Canny operator, while other operators gave similar, but lower values. In other cases, the proposed algorithm records substantially better FoM values. For images with a medium number of details, the best edge detection enhancements have been achieved in the third level using the proposed algorithm.

Since the compression ratio is higher in the image with a high number of details, based on this fact, it can be concluded that the detection will be much worse. Also, in the case of images with a high number of details, the best improvements are achieved with the Canny operator at all levels. Based on the results obtained, it can be seen that the best improvements are achieved in the third level.

3.3 Performance Ratio

Tables 8, 9 and 10 show PR values before and after applying the proposed approach using five edge detection operators and three wavelet decomposition levels. Table 8 shows the values for an image with a low number of details. Based on the results obtained, it appears that improvements have been made with certain operators. However, based on the values obtained in Table 8, it can be seen that the old PR values and new PR values obtained using the proposed approach have greatest difference when using the Robert operator.

Table 9 contains the PR values obtained for an image with medium number of details. From Table 9 can be seen that the new values obtained are best in the second and third decomposition levels. In other words, the best improvements are achieved in the second and third levels, depending on the operator used and the order of the db wavelet.

Table 10 contains PR values obtained for an image with a high number of details. Based on these results, it can be seen that improvements have been achieved using the proposed approach, especially in the third decomposition level.

Wavelet		**db2**		**db4**		**db6**		**db8**		**db10**	
	Operator	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**
Level1	**Canny**	88.955	**91.928**	89.461	89.272	85.982	**91.960**	86.025	**86.914**	88.767	**89.626**
	LoG	89.163	89.024	90.209	**90.514**	86.915	85.582	83.605	**85.108**	89.779	88.556
	Prewitt	91.784	**91.937**	92.341	92.327	92.358	**92.572**	92.664	92.632	92.868	92.540
	Sobel	91.891	91.712	92.678	**92.695**	92.423	**92.466**	93.031	92.934	92.891	92.742
	Roberts	95.150	**95.387**	89.419	**95.816**	86.680	**95.561**	93.837	95.613	83.259	**95.653**
Level2	**Canny**	61.727	**66.006**	51.308	**52.452**	40.207	**41.320**	31.974	**34.924**	36.049	**37.174**
	LoG	75.006	**79.220**	60.472	**66.126**	47.187	**51.753**	45.955	**50.186**	44.379	**49.053**
	Prewitt	82.182	76.052	88.824	**92.094**	88.654	**91.490**	89.074	**91.115**	88.676	**91.615**
	Sobel	82.785	77.091	88.818	**91.599**	88.581	**91.498**	88.979	**90.786**	88.716	**91.575**
	Roberts	82.494	**85.980**	89.832	88.389	91.016	88.300	91.281	**91.386**	91.877	**92.731**
Level3	**Canny**	6.045	**9.491**	34.316	**38.640**	27.035	**34.575**	25.158	**33.672**	26.926	**33.476**
	LoG	42.385	**48.432**	49.201	**53.907**	47.312	**50.844**	48.132	**51.458**	48.646	**52.246**
	Prewitt	52.036	**55.246**	78.675	**71.700**	78.136	**85.231**	76.196	**85.186**	75.841	**84.392**
	Sobel	52.113	**55.991**	78.606	**73.264**	78.088	**86.561**	76.190	**84.761**	75.837	**84.193**
	Roberts	57.384	**74.222**	78.821	**81.909**	77.951	**81.587**	78.839	**82.599**	78.901	**82.152**

Table 5. *FoM* values for an image with a low number of details (LD)

Wavelet		**db2**		**db4**		**db6**		**db8**		**db10**	
	Operator	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**
Level1	**Canny**	83.334	**82.917**	83.593	**83.835**	83.633	**83.926**	83.596	**83.773**	83.715	**84.068**
	LoG	76.358	76.351	76.984	76.892	77.003	76.875	77.216	76.957	77.230	77.050
	Prewitt	65.096	64.526	63.403	63.351	63.694	63.564	63.058	62.834	63.696	63.335
	Sobel	65.061	64.397	65.553	63.420	63.948	63.449	63.076	62.941	63.705	63.444
	Roberts	58.484	**58.800**	60.477	58.903	56.160	55.423	58.046	56.431	57.563	56.511
Level2	**Canny**	80.273	**80.650**	80.260	**80.889**	80.963	**81.348**	80.921	**81.384**	81.200	**81.295**
	LoG	73.433	73.272	73.300	**73.418**	73.474	**73.511**	73.261	73.002	73.519	73.504
	Prewitt	60.830	60.702	55.991	**56.815**	55.317	**56.128**	54.346	**55.447**	55.118	56.228
	Sobel	60.843	60.732	56.057	**56.975**	55.354	**56.375**	54.463	**55.598**	55.177	**56.389**
	Roberts	54.327	50.858	50.265	**51.582**	46.033	**46.891**	43.300	**43.996**	43.972	**44.406**
Level3	**Canny**	70.628	**75.211**	68.835	**73.034**	68.847	**73.556**	66.794	**71.193**	66.968	**71.698**
	LoG	61.503	**61.749**	51.323	**53.911**	44.316	**47.050**	41.310	**43.968**	39.284	**41.870**
	Prewitt	49.680	**50.908**	40.509	**44.000**	36.667	**40.527**	34.840	28.934	35.081	**38.936**
	Sobel	49.618	**51.109**	40.543	**43.917**	36.654	**40.182**	34.910	38.611	35.098	**38.686**
	Roberts	42.298	38.790	31.990	**34.236**	19.143	**20.251**	15.979	**16.983**	16.216	**17.198**

Table 6. FoM values for an image with a medium number of details (MD)

Wavelet		**db2**		**db4**		**db6**		**db8**		**db10**	
	Operator	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**
Level1	**Canny**	70.111	**71.230**	69.127	**71.555**	67.787	**70.466**	68.057	**69.417**	68.420	**70.459**
	LoG	80.612	80.029	80.713	80.131	81.016	80.593	80.894	80.399	81.020	80.543
	Prewitt	64.338	63.895	58.063	**58.180**	62.165	61.852	59.804	**59.969**	60.513	60.454
	Sobel	64.259	63.817	58.435	**58.711**	62.340	62.180	59.992	**60.167**	60.924	60.659
	Roberts	52.692	52.351	56.121	53.588	49.446	47.979	53.204	51.154	49.377	47.762
Level2	**Canny**	79.241	**79.417**	80.766	**80.868**	79.811	**79.977**	80.195	**80.628**	79.964	**81.130**
	LoG	74.034	73.853	74.988	74.765	74.498	74.087	73.826	73.502	75.316	74.986
	Prewitt	57.164	56.225	51.862	51.670	50.198	**51.235**	47.020	**47.824**	50.519	**51.654**
	Sobel	57.186	56.503	51.594	51.833	50.409	**51.258**	47.040	**47.927**	50.545	**51.554**
	Roberts	48.516	45.265	45.511	**46.339**	42.075	41.858	37.190	**37.934**	39.990	**40.475**
Level3	**Canny**	69.148	**73.049**	66.302	**70.065**	67.704	**71.391**	66.096	**69.790**	66.069	**70.312**
	LoG	61.954	**63.395**	51.619	**54.322**	45.840	**48.129**	41.824	**44.563**	39.693	**41.655**
	Prewitt	48.604	**50.094**	36.487	**39.421**	32.028	**35.209**	29.176	**32.407**	29.650	**33.184**
	Sobel	48.670	**50.230**	36.407	**39.104**	32.080	**35.073**	29.309	**32.294**	29.656	**32.994**
	Roberts	41.724	37.448	29.368	**31.066**	21.038	**21.889**	19.951	**20.950**	20.829	**21.832**

Table 7. FoM values for an image with a high number of details (HD)

Wavelet		**db2**		**db4**		**db6**		**db8**		**db10**	
	Operator	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**
Level1	**Canny**	28.695	**30.234**	30.519	28.649	29.164	**30.704**	28.526	**28.800**	29.527	**29.945**
	LoG	22.532	**22.923**	22.798	**23.225**	22.589	21.305	21.473	**21.741**	22.573	**23.038**
	Prewitt	28.367	26.024	27.553	26.478	28.856	28.676	23.739	**26.379**	30.706	28.426
	Sobel	28.795	25.315	27.667	26.757	29.537	28.484	24.113	**26.399**	30.201	28.388
	Roberts	57.813	**64.519**	50.543	**68.677**	53.857	**67.983**	61.243	**65.303**	52.937	**69.606**
Level2	**Canny**	19.222	**21.983**	16.320	**18.378**	12.742	**14.538**	11.667	**12.021**	12.019	**13.333**
	LoG	17.165	**18.843**	15.345	**17.218**	11.803	**13.387**	12.855	**13.068**	11.443	**12.642**
	Prewitt	18.380	**21.641**	19.978	**25.675**	19.899	**23.619**	25.535	21.490	19.300	**26.996**
	Sobel	17.914	**22.085**	19.811	**23.516**	19.587	**22.875**	24.922	21.730	19.224	**24.930**
	Roberts	21.227	**35.587**	26.948	**41.347**	32.925	**41.833**	40.560	**47.205**	38.156	**48.936**
Level3	**Canny**	6.045	**9.491**	6.356	**10.141**	5.286	**10.217**	5.549	**9.375**	5.909	**10.356**
	LoG	7.358	**10.470**	8.717	**11.731**	7.600	**11.502**	8.702	**10.541**	48.646	**52.246**
	Prewitt	8.616	**12.775**	11.630	**15.895**	11.256	**16.399**	11.010	**16.723**	12.019	**17.248**
	Sobel	8.517	**12.598**	11.485	**14.844**	11.135	**15.399**	10.871	**15.795**	12.077	**16.171**
	Roberts	11.221	**19.060**	13.107	**17.118**	13.978	**21.112**	17.214	**25.186**	18.270	**26.741**

Table 8. PR values for an image with a low number of details (LD)

Wavelet		**db2**		**db4**		**db6**		**db8**		**db10**	
	Operator	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**
Level1	**Canny**	23.355	23.137	23.791	23.282	23.609	**23.231**	23.603	23.428	23.612	23.480
	LoG	15.246	**15.296**	15.399	**15.421**	15.350	**15.459**	15.426	15.413	15.538	15.431
	Prewitt	14.814	14.285	14.024	**14.098**	14.485	14.477	13.920	**14.028**	14.634	14.595
	Sobel	14.872	14.320	14.108	14.003	14.558	**14.568**	13.888	**14.006**	14.651	14.536
	Roberts	16.694	16.557	18.245	**18.317**	18.306	17.373	18.887	18.219	19.595	18.454
Level2	**Canny**	20.906	**21.648**	21.376	**22.659**	21.166	**22.678**	21.246	**22.690**	21.387	**22.407**
	LoG	13.539	13.402	13.020	**13.546**	13.366	**13.549**	13.234	**13.610**	13.059	**13.478**
	Prewitt	12.883	**13.215**	10.217	**11.158**	9.525	**11.262**	9.404	**11.097**	9.781	**11.356**
	Sobel	12.865	**13.208**	10.178	**11.005**	9.568	**11.110**	9.390	**10.842**	9.730	**11.182**
	Roberts	9.013	**9.132**	9.337	**12.206**	9.369	**11.179**	9.167	**10.777**	9.362	**10.936**
Level3	**Canny**	11.428	**15.604**	68.835	**73.034**	13.660	**16.821**	12.749	**16.686**	12.776	**16.643**
	LoG	7.454	**9.010**	6.240	**7.788**	4.806	**6.272**	4.410	**5.796**	4.104	**5.568**
	Prewitt	6.705	**8.030**	5.085	**6.670**	4.245	**6.477**	3.867	**6.152**	4.040	**6.210**
	Sobel	6.700	**7.825**	5.088	**6.452**	4.237	**6.091**	3.870	**5.821**	4.040	**5.867**
	Roberts	4.250	**4.427**	3.472	**4.805**	2.185	**2.855**	1.765	**2.435**	1.816	**2.471**

Table 9. PR values for an image with a medium number of details (MD)

Wavelet		**db2**		**db4**		**db6**		**db8**		**db10**	
	Operator	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**
Level1	**Canny**	70.111	**71.230**	69.127	**71.555**	67.787	**70.466**	68.057	**69.417**	68.420	**70.459**
	LoG	80.612	80.029	80.713	80.131	81.016	80.593	80.894	80.399	81.020	80.543
	Prewitt	64.338	63.895	58.063	**58.180**	62.165	61.852	59.804	**59.969**	60.513	60.454
	Sobel	64.259	63.817	58.435	**58.711**	62.340	62.180	59.992	**60.167**	60.924	60.659
	Roberts	52.692	52.351	56.121	53.588	49.446	47.979	53.204	51.154	49.377	47.762
Level2	**Canny**	79.241	**79.417**	80.766	**80.868**	79.811	**79.977**	80.195	**80.628**	79.964	**81.130**
	LoG	74.034	73.853	74.988	74.765	74.498	74.087	73.826	73.502	75.316	74.986
	Prewitt	57.164	56.225	51.862	51.670	50.198	**51.235**	47.020	**47.824**	50.519	**51.654**
	Sobel	57.186	56.503	51.954	51.833	50.409	**51.258**	47.040	**47.927**	50.545	**51.554**
	Roberts	48.516	45.265	45.511	**46.339**	42.075	41.858	37.190	**37.934**	39.990	**40.475**
Level3	**Canny**	69.148	**73.049**	66.302	**70.065**	67.704	**71.391**	66.096	**69.790**	66.069	**70.312**
	LoG	61.954	**63.395**	51.619	**54.322**	45.840	**48.129**	41.824	**44.563**	39.693	**41.655**
	Prewitt	48.604	**50.094**	36.487	**39.421**	32.028	**35.209**	29.176	**32.407**	29.650	**33.184**
	Sobel	48.670	**50.230**	36.407	**39.104**	32.080	**35.073**	29.309	**32.294**	29.656	**32.994**
	Roberts	41.724	**37.448**	29.368	**31.066**	21.038	**21.889**	19.951	**20.950**	20.829	**21.832**

Table 10. PR values for an image with a high number of details (HD)

4 CONCLUSIONS

This paper proposed a new approach to edge detection in the images on which the wavelet decomposition was applied to the third level. As mother wavelet, Daubechies was used from the second to the tenth order. The analyzed images are categorized into three complexity criteria, so they consist of a small, medium and high number of details. F measure, FoM, and PR were used for an objective measure. The proposed approach provides significant improvements in edge detection for almost all operators (Canny, LoG, Prewitt, Sobel).

Depending on the number of details in the image, the decomposition level as well as db wavelet, the improvements are different. With a small number of details, the greatest improvements were achieved with the Canny operator. Other operators also achieved improvement but depending on the db wavelet order. Based on the obtained results, in the image with a small number of details, it can be seen that best improvements are achieved in the third level using Laplacian operators. In the image with the medium number of details, in the first level similar results are generally obtained, while in the second and the third levels improvement is made using the proposed approach. For images with a high number of details, better or similar values are obtained using the proposed approach. What can be concluded is that in the second level, the best values are obtained by gradient operators.

Considering that the compression ratio is higher of the image with a higher number of details, based on this fact, it can be concluded that the detection will also be poorer, and consequently there will be a lower F values, FoM values and PR values. Since the proposed approach is intended for systems where compression is used, i.e. compressed image processing, it can be concluded that increasing the degree of compression also provides a better difference, or better value using the proposed approach.

Today's systems require image quality to be as good as possible, with as much compression as possible in order to process these images in real time, such as edge detection, segmentation, streaming, streaming in systems using augmented reality, etc. The proposed approach can find many practical applications in all systems where real-time information needs to be processed, especially in television systems, but it also provides a good basis for the direction of future research related to compression and edge detection.

Acknowledgment

This work was done within the research project of the Ministry of Science and Technological Development of Serbia TR35026 and III47016.

REFERENCES

[1] Chen, Y.—Wang, D.—Bi, G.: An Image Edge Recognition Approach Based on Multi-Operator Dynamic Weight Detection in Virtual Reality Scenario. Cluster Computing, Vol. 22, 2018, Suppl. 4, pp. 8069–8077, doi: 10.1007/s10586-017-1604-y.

[2] Srivastava, D.—Kohli, R.—Gupta, S.: Implementation and Statistical Comparison of Different Edge Detection Techniques. In: Bhatia, S., Mishra, K., Tiwari, S., Singh, V. (Eds.): Advances in Computer and Computational Sciences. Springer, Singapore, Advances in Intelligent Systems and Computing, Vol. 553, 2017, pp. 211–228, doi: 10.1007/978-981-10-3770-2_20.

[3] Maini, R.—Aggarwal, H.: Study and Comparison of Various Image Edge Detection Techniques. International Journal of Image Processing, Vol. 3, 2009, No. 1, pp. 1–11.

[4] Chaganti, V. R.: Edge Detection of Noisy Images Using 2-D, Discrete Wavelet Transform. Thesis, Florida State University Libraries, 2005.

[5] Dong, X.—Li, M.—Miao, J.—Wang, Z.: Edge Detection Operator for Underwater Target Image. 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), 2018, pp. 91–95, doi: 10.1109/ICIVC.2018.8492749.

[6] Kuang, T.—Zhu, Q-X.—Sun, Y.: Edge Detection for Highly Distorted Images Suffering Gaussian Noise Based on Improve Canny Algorithm. Kybernetes, Vol. 40, 2011, No. 5-6, pp. 883–893, doi: 10.1108/03684921111142430.

[7] Jun, L.: A Wavelet Approach to Edge Detection. Thesis, Sam Houston State University, Huntsville, Texas, 2003.

[8] Muthukrishnan, R.—Radha, M.: Edge Detection Techniques for Image Segmentation. International Journal of Computer Science and Information Technology, Vol. 3, 2011, No. 6, pp. 259–267, doi: 10.5121/ijcsit.2011.3620.

[9] Yang, T.—Sun, G.—Duan, X.: A New Method of Wavelet Transform-Based Edge Detection. 2011 IEEE 13th International Conference on Communication Technology, Jinan, 2011, pp. 789–792, doi: 10.1109/ICCT.2011.6157985.

[10] Yelampalli, P. K. R.—Nayak, J.—Gaidhane, V. H.: Daubechies Wavelet-Based Local Feature Descriptor for Multimodal Medical Image Registration. IET Image Processing, Vol. 12, 2018, No. 10, pp. 1692–1702, doi: 10.1049/iet-ipr.2017.1305.

[11] Wang, S.-H.—Zhang, Y.-D.—Dong, Z.—Phillips, P.: Wavelet Families and Variants. Chapter 6. In: Wang, S.-H. et al.: Pathological Brain Detection. Springer, Singapore, Brain Informatics and Health, 2018, pp. 85–104, doi: 10.1007/978-981-10-4026-9_6.

[12] Balachandran, A.—Ganesan, M.—Sumesh, E. P.: Daubechies Algorithm for Highly Accurate ECG Feature Extraction. 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE), Coimbatore, pp. 1–5, 2014, doi: 10.1109/ICGCCEE.2014.6922266.

[13] Rana, K.—Thakur, S.: Comparisons of Wavelets and Algorithms Based on Wavelets and Comparing the Results with JPEG. 2017 International Conference on

Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, 2017, pp. 3871–3876, doi: 10.1109/ICECDS.2017.8390189.

[14] Zhao, D.—Sun, G.: A Novel Edge Detection Approach Using Multi-Resolution Image. 2nd International Conference on Computer Engineering and Technology, Chengdu, 2010, pp. V2-692–V2-695, doi: 10.1109/ICCET.2010.5485706.

[15] Rao, G. S. B.—Prasad, P. M. K.—Kumar, M. N. V. S. S.: Investigation of Various Orthogonal Wavelets for Precise Analysis of X-Ray Images. Journal of Engineering Research and Applications, Vol. 5, 2015, No. 2, Part 3, pp. 24–32.

[16] Ilic, S.—Petrovic, M.—Jaksic, B.—Spalevic, P.—Lazic, L.—Milosevic, M.: Experimental Analysis of Picture Quality after Compression by Different Methods. Przegląd Elektrotechniczny, Vol. 89, 2013, No. 11, pp. 190–194.

[17] Delac, K.—Grgic, M.—Grgic, S.: Effects of JPEG and JPEG2000 Compression on Face Recognition. In: Singh, S., Singh, M., Apte, C., Perner, P. (Eds.): Pattern Recognition and Image Analysis (ICAPR 2005). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 3687, 2005, pp. 136–145, doi: 10.1007/11552499_16.

[18] Rani, K. S.—Rao, D. N.: A Comparative Study of Various Noise Removal Techniques Using Filters. Research and Reviews: Journal of Engineering and Technology, Vol. 7, 2018, No. 2, pp. 47–52.

[19] Wen, X.—Deng, Z.—Xue, H.: An Improved Median Filtering Image De-Noising Algorithm. Revista Tecnica de la Facultad de Ingenieria Universidad del Zulia, Vol. 39, 2016, No. 5, pp. 293–298, doi: 10.21311/001.39.5.38.

[20] The Berkeley Segmentation Dataset and Benchmark. Available at: `https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/`, accessed: 14.02.2019.

[21] Arbeláez, P.—Maire, M.—Fowlkes, C.—Malik, J.: Contour Detection and Hierarchical Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 33, 2011, No. 5, pp. 898–916, doi: 10.1109/TPAMI.2010.161.

[22] Hagara, M.—Kubinec, P.: About Edge Detection in Digital Images. Radioengineering, Vol. 27, 2018, No. 4, pp. 919–929, doi: 10.13164/re.2018.0919.

[23] Khaire, P. A.—Thakur, N. V.: A Fuzzy Set Approach for Edge Detection. International Journal of Image Processing (IJIP), Vol. 6, 2012, No. 6, pp. 403–412.

[24] Jabbar, S. I.—Day, C. R.—Heinz, N.—Chadwick, E. K.: Using Convolutional Neural Network for Edge Detection in Musculoskeletal Ultrasound Images. Proceeding of the 2016 International Joint Conference on Neural Networks (IJCNN), pp. 4619–4626, 2016, doi: 10.1109/IJCNN.2016.7727805.

Vladimir Maksimović received his B.Sc. and M.Sc. degrees in electrical engineering from the Faculty of Technical Sciences in Kosovska Mitrovica, University of Priština, Serbia. He is Ph.D. candidate at the same faculty and his areas of research include image processing, telecommunications and multimedia systems. He has authored several scientific papers.

Branimir Jakšić received his B.Sc. and M.Sc. degrees in electrical engineering from the Faculty of Technical Sciences in Kosovska Mitrovica, University of Priština, Serbia, and his Ph.D. degree in electrical engineering from the Faculty of Electronic Engineering, University of Niš, Serbia in 2015. He is Assistant Professor at the Faculty of Technical Sciences in Kosovska Mitrovica. Areas of his research include telecommunications and multimedia systems. He has authored over 70 scientific papers on the above subject.

Mile Petrović is Full Professor at the Department of Electronics and Telecommunications Engineering at the Faculty of Technical Sciences in Kosovska Mitrovica, Serbia. He has extensive experience in digital broadcasting systems and multimedia applications, mobile technology and digital image processing. He is the author of over 100 scientific peer-reviewed papers, initiator of a number of international Tempus projects in the country and certified patents. He has participated in scientific research projects as well as in projects for the modernization and enhancing quality of higher education. He is author of many textbooks and scientific articles in the field of mobile radio and telecommunication networks.

Petar Spalević received his B.Sc. degree from the Faculty of Electronic Engineering, University of Priština, in 1997 and his M.Sc. and Ph.D. degrees from the Faculty of Electronic Engineering, University of Niš in 1999 and 2003, respectively. He is Professor at the Faculty of Technical Sciences, Department of Telecommunications in Kosovska Mitrovica. His primary research interests are statistical communications theory, optical and wireless communications, applied probability theory and optimal receiver design.

Stefan Panić received his M.Sc. and Ph.D. degrees in electrical engineering from the Faculty of Electronic Engineering, Niš, Serbia, in 2007 and 2010, respectively. His research interests in mobile and multi-channel communications include statistical characterization and modelling of fading channels, performance analysis of diversity combining techniques, outage analysis of multi-user wireless systems subject to interference. Within digital communication, his current research interests include the information theory, source and channel coding, and signal processing. He has published over 40 SCI indexed papers. Currently he is Associated Professor at the Department of Informatics, Faculty of Natural Science and Mathematics, University of Priština.

Computing and Informatics, Vol. 38, 2019, 1091–1110, doi: 10.31577/cai_2019_5_1091

OVERLAPPING COMMUNITY DETECTION EXTENDED FROM DISJOINT COMMUNITY STRUCTURE

Yan Xing

School of Computer Science and Technology
Civil Aviation University of China
Tianjin, 300300, China
&
School of Computer Science and Technology
China University of Mining and Technology
Xuzhou, Jiangsu, 221116, China
e-mail: `yxing425@163.com`

Fanrong Meng, Yong Zhou, Guibin Sun, Zhixiao Wang*

School of Computer Science and Technology
China University of Mining and Technology
Xuzhou, Jiangsu, 221116, China
e-mail: `{mengfr, yzhou}@cumt.edu.cn, sunguibinbest@qq.com,`
`zhxwang@cumt.edu.cn`

Abstract. Community detection is a hot issue in the study of complex networks. Many community detection algorithms have been put forward in different fields. But most of the existing community detection algorithms are used to find disjoint community structure. In order to make full use of the disjoint community detection algorithms to adapt to the new demand of overlapping community detection, this paper proposes an overlapping community detection algorithm extended from disjoint community structure by selecting overlapping nodes (ONS-OCD). In the algorithm, disjoint community structure with high qualities is firstly taken as input, then, potential members of each community are identified. Overlapping nodes are determined according to the node contribution to the community. Finally, adding

* Corresponding author

overlapping nodes to all communities they belong to and get the final overlapping community structure. ONS-OCD algorithm reduces the computation of judging overlapping nodes by narrowing the scope of the potential member nodes of each community. Experimental results both on synthetic and real networks show that the community detection quality of ONS-OCD algorithm is better than several other representative overlapping community detection algorithms.

Keywords: Disjoint community detection, overlapping community detection, potential member, overlapping node

Mathematics Subject Classification 2010: 68-Q87

1 INTRODUCTION

Complex network is a relatively stable relation system which is formed by the interaction between individual members. Many real-world complex systems can be described by the form of complex networks, such as social networks, scientists cooperation networks, web networks, protein interaction networks, etc. [1]. Extensive studies have shown that complex networks not only have the properties of small world [2] and scale-free [3], but they also have the characteristic of community (module or cluster) structure. A community in a network is a group of nodes with dense connections within the group and only sparse connections between them [4]. Research on community detection of complex networks has important theoretical significance and wide application prospect. Community detection in complex networks can help to explore the structure and function of the network, find the hidden laws and predict their behavior [1]. Therefore, community detection is the basis and key of network analysis.

Traditional community detection algorithms divide the network into a number of disjoint communities. Each node can only belong to one community. Representative methods include modularity optimization algorithms [5, 6, 7], spectral clustering algorithms [8, 9], hierarchical partition algorithms [10, 11], label propagation based algorithms [12, 13], information theory based algorithms [14], and so forth. However, in many real complex networks, communities are usually not isolated from each other, but overlap and cross each other. Some nodes may belong to many communities at the same time. For example, a researcher may belong to different research groups. Therefore, finding overlapping community structure in complex networks has more practical significance.

Currently, the research on overlapping community detection has attracted more and more attention. After the development of the past few years, there have been a number of algorithms to detect overlapping communities. For example, the clique percolation method (CPM) [15], algorithms based on local community optimization and expansion (LFM [16], OSLOM [17], DEMON [18], etc.), multi label propagation algorithms (COPRA [19], BMLPA [20], SLPA [21], etc.), algorithms based on link clustering (LINK [22], LinkComm [23], LGPSO [24], LLCM [25], LBLP [26], GaoCD [27], etc.). But the computational complexity of these algorithms is generally very high while the accuracy and stability is low. The research on disjoint community detection has reached a higher level in the past decades, and some high quality algorithms in terms of both computational complexity and accuracy have been developed. By contrast, the development of overlapping community detection is not enough.

In most cases, disjoint community structures with high qualities already contain the basic and major community structure in the network, except the overlapping part [28]. On this basis, we only need to further identify the overlapping nodes in the community. Overlapping node detection can help us to understand the characteristics of nodes more comprehensively and plays a key role in community evolution. Literature [29] proposed a new algorithm based on disjoint community detection results. Firstly, the border nodes of each community are detected according to the results of disjoint communities. Then the impact of these border nodes on the corresponding community is analyzed. If the impact value is greater than 0, the border node is added into this new community and remains in the original communities. Otherwise, the border node is removed from this community. Finally, the overlapping community structure is obtained. OCDBIDC [30] is also based on the results of disjoint community detection. But it only adds boundary nodes which increase the boundary sharpness of a community into the community.

Inspired by these, this paper proposes an overlapping community detection algorithm extended from disjoint community by selecting overlapping nodes, named ONS-OCD. Firstly, ONS-OCD determines potential members of each community based on the given disjoint community structure. According to the optimization theory, if the quality of the division is already high, then the addition of a new node will not obviously change the intensity of the community. Thus, we can get two conditions to judge whether a node is a potential member of a community. One is that it should be the external fringe node of the community, namely that there are edges between the node and the internal nodes of this community. Another is that the similarity between the node and the community should be larger than the given threshold. Then, ONS-OCD detects the overlapping nodes to get the overlapping community structure. It analyses every single potential node of each community. If the influence of the potential node on the community is larger than zero, we add this node to the community and mark it as an overlapping node.

The main idea behind ONS-OCD and its contributions are presented below:

1. ONS-OCD firstly finds the potential members of each community to reduce the detection scope of overlapping nodes;
2. ONS-OCD uses the node similarity based on the heuristic DFS encoding which is more precise to measure the relationship between nodes.

We verify the performance of the proposed algorithm on synthetic and real networks. Extensive experimental studies confirm that ONS-OCD can detect overlapping community structures more effectively compared with some other state-of-the-art algorithms.

The rest of this paper is organized as follows: Section 2 introduces the basic theories related to this paper. In Section 3, we describe the main idea of the proposed algorithm. The experimental results on both synthetic and real networks in Section 4 demonstrate the effectiveness of the proposed algorithm. The conclusion is given in Section 5.

2 BACKGROUND AND RELATED WORKS

2.1 Representation of Complex Network

A complex network can be modeled as a graph $G = (V, E)$, where $V = \{v_1, v_2, \dots, v_n\}$ is the set of nodes and $E = \{e_1, e_2, \dots, e_m\}$ is the set of edges, n and m are the number of nodes and edges in the network. $N(v_u)$ represents the neighbor set of node v_u and $Com(v_u)$ represents the community set which node v_u belongs to. $C = \{C_1, C_2, \dots, C_k\} (1 < k < n)$ is the set of community structures, where $C_i \in C$ is a nonempty subset of V and the union of all communities are the union of all nodes in the network, $\bigcup_{i=1}^{k} C_k = V$.

Disjoint community detection algorithms divide the nodes of the network into some non-overlapping subsets. That is to say each node must belong to only one community and the intersection of any two communities is empty, $C_i \bigcap C_j = \Phi$, $i, j = 1, 2, \dots, k$ and $i \neq j$. While overlapping community detection algorithms allow nodes to belong to one or more communities.

2.2 Node Structural Similarity

Structural similarity is a commonly used method for measuring the node similarity in complex networks. There are many methods to compute the structural similarity and these methods determine node similarity based solely on the structure of the network. Since structural equivalence is too restrictive for practical use, some simplified similarity measures can be used [31]. Here we introduce the cosine similarity. If the node v_u and node v_w are connected, the structural similarity of the node v_u and node v_w is represented as $\mathrm{Scosine}(v_u, v_w)$ and calculated by Equation (1).

$$\mathrm{Scosine}(v_u, v_w) = \frac{|N(v_u) \bigcap N(v_w)|}{\sqrt{|N(v_u)||N(v_w)|}}. \tag{1}$$

The structural similarity between two nodes represents the degree of their shared neighbors.

2.3 DFS Encoding of Nodes

Depth first search (DFS) encoding [32] is a repeated random process based on the DFS for the graph. In each process, the DFS is started from a randomly selected node and each node is re-encoding by DFS traversal order. The coding of node v_u is marked as $DFS(v_u)$. Thus, for any two nodes v_u and v_w, the absolute difference between their coding indicates the distance of these two nodes, denoted as $dis(v_u, v_w) = |DFS(v_u) - DFS(v_w)|$. The similarity between nodes v_u and v_w is represented by the reciprocal of the distance between them, $s(v_u, v_w) = 1/dis(v_u, v_w)$. Repeat the random process many times, and average these similarities between node v_u and v_w as the final node similarity $S_{DFS}(v_u, v_w)$.

2.4 Node Similarity Based on Heuristic DFS Encoding

DFS encoding is a depth first search process starting from a random node and encoding each node based on the traversal order. In this paper, the heuristic rules of heuristic DFS (HDFS) encoding guides the DFS process to traverse the nodes in the same community firstly. That is to say, in the traversal process, the node which has the maximum structural similarity with the current expansion node is always firstly chosen to be traversed. For any two nodes, if their values of HDFS encoding are close, the similarity between them is large.

Since HDFS encoding has some randomness, the node similarity is calculated by using the average value of multiple HDFS encoding. For any two nodes v_u and v_w, the similarity based on HDFS encoding is denoted as $S_{HDFS}(v_u, v_w)$. In this paper, the execution number of HDFS encoding is set to be the number of communities in the network, and in each process, the node with the largest node degree of each community is selected as the initial expanding node.

3 OVERLAPPING NODE SELECTION METHOD

We propose an overlapping node selection method based on the disjoint community structure. In order to better understand the algorithm model, we first introduce a few definitions, and then detailedly introduce the process of the algorithm proposed in this paper.

3.1 Related Definitions

Definition 1 (Similarity between node and community)**.** The maximum similarity between the node and the community members is the similarity between the node and the community. The similarity between node v_u and the community C_i is denoted as $\mathrm{SNC}(v_u, C_i)$ and calculated by Equation (2).

$$\mathrm{SNC}(v_u, C_i) = \max_{v_w \in C_i} S(v_u, v_w) \tag{2}$$

where $S(v_u, v_w)$ is the similarity between node v_u and node v_w and any kind of node similarity in complex network can be used in this formula. In this paper, we choose the node similarity based on HDFS coding to measure the similarity between any two nodes. Here $S(v_u, v_w) = S_{HDFS}(v_u, v_w)$.

Definition 2 (The potential member of a community)**.** For a community, the nodes in the network can be divided into three classifications: external nodes, internal nodes and fringe nodes. External node means a node outside of the community; internal node is the node which is within the community and is not connected with the external node; fringe node is within the community and connected with the external node. In Figure 1, the area of I is the internal node of community C, B is the fringe node of the community C, U is the external node of community C. The external nodes can be further divided into true external nodes and external fringe nodes. The external fringe node is the node which is outside the community and is connected with the fringe node, represented by UB.

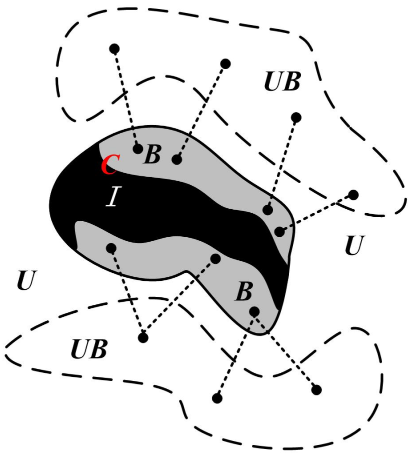


Figure 1. Node classification in complex networks

The potential member of the community needs to meet two conditions at the same time. It must be the external fringe node of the community and the similarity between the node and the community is greater than a given threshold (the threshold value can be set to the similarity between the node and the current community it belongs to).

Definition 3 (Community strength)**.** Based on the theory that the similarity between the nodes in the same community should be as large as possible, and the nodes in different communities should be as different as possible, we define the community strength as the ratio of the sum similarity between internal nodes of the community and their adjacent nodes within the community to the sum similarity between internal nodes of the community and all their adjacent nodes in the networks. The larger the ratio is, the more obvious the community structure is and the greater the community strength is. Community strength calculation formula is shown as Equation (3).

$$R(C_i) = \frac{\sum_{v_u \in C_i} \sum_{v_w \in C_i, v_w \in N(v_u)} S_{HDFS}(v_u, v_w)}{\sum_{v_u \in C_i} \sum_{v_w \in N(v_u)} S_{HDFS}(v_u, v_w)} \tag{3}$$

where $\sum_{v_w \in C_i, v_w \in N(v_u)} S_{DHFS}(v_u, v_w)$ is the sum of similarity between node v_u and all its neighbor nodes within the community C_i and $\sum_{v_w \in N(v_u)} S_{DHFS}(v_u, v_w)$ is the sum of similarity between node v_u and all its neighbor nodes in the networks.

Definition 4 (The influence of node on community)**.** The variation of the community strength before and after the node joins the community is the influence of the node on the community. The calculation of the influence of the node v_u on the community C_i, denoted as $F(C_i, v_u)$, is shown as Equation (4).

$$F(C_i, v_u) = R(C_i \bigcup \{v_u\}) - R(C_i \backslash \{v_u\}). \tag{4}$$

Definition 5 (Overlapping node)**.** If a node belongs to more than one community at the same time, it is an overlapping node. That is to say, if $|Com(v_u)| > 1$, node v_u is an overlapping node.

3.2 Pseudo Code of the Algorithm

ONS-OCD contains two stages. The first stage is to find the potential members of each community and construct the potential node set (PNS) of each community. The second stage is to analyze the potential member nodes and get the set of overlapping nodes (ONS). In the first stage, ONS-OCD selects the external fringe node of the community. Then, it determines whether the node is a potential member of the community according to the similarity between the node and the community, and obtains the potential members of the node set PNS (line 7–9). In the second stage, ONS-OCD traverses PNS set of each community and calculates the influence of every node on the community. If the influence of node v_u on the community is positive, node v_u is added to the community and becomes an overlapping node (line 15–24).

3.3 Time Complexity Analysis

Assuming that the network G contains n nodes and m edges, the time complexity analysis of the improved algorithm proposed in this paper is as follows:

1. Compute the HDFS similarity: The time complexity of HDFS is $O(m)$, and it is repeated k times, where k is the number of communities in the network and $k << n$. So the time complexity is $O(km)$;

Algorithm 1 Overlapping community detection extended from disjoint community structure (ONS-OCD)

```
Input:  $G = (V, E)$, disjoint community structure $DC = \{DC_1, DC_2, \ldots, DC_k\}$
Output: overlapping community structure $OC = \{OC_1, OC_2, \ldots, OC_k\}$
 1: // The first stage, find the potential members
 2: for each $DC_i \in DC$ do
 3:     $PNS[i] \leftarrow \Phi$
 4:     for each $v_u \in DC_i$ do
 5:         for each $v_w \in N(v_u)$ do
 6:             if $v_w \notin DC_i$ and $v_w \in DC_j$ then
 7:                 if $\mathrm{SNC}(v_w, DC_i) > \mathrm{SNC}(v_w, DC_j)$ then
 8:                     $PNS[i] \leftarrow PNS[i] \bigcup \{v_w\}$
 9:                 end if
10:             end if
11:         end for
12:     end for
13: end for
14: // The second stage, find the overlapping nodes
15: for each $PNS[i] \in PNS$ do
16:     $ONS_i \leftarrow \Phi$
17:     $OC_i \leftarrow DC_i$
18:     for each $v_u \in PNS[i]$ do
19:         if $F(DC_i, v_u) > 0$ then
20:             $OC_i \leftarrow OC_i \bigcup \{v_u\}$
21:             $ONS_i \leftarrow ONS_i \bigcup \{v_u\}$
22:         end if
23:     end for
24: end for
25: return $OC$
```

2. Judge the community potential node: $O(nd)$, where d is the average degree of nodes in the network;
3. Judge the overlapping node: $O(n'd)$, where n' is the number of potential nodes and $n' << n$.

The time complexity is $O(km) + O(nd) + O(n'd)$, taking into account that in many real networks k, n' and d are much less than n, m has the linear relationship with n, therefore, the overall time complexity of ONS-OCD is $O(m)$ or $O(n)$.

4 EXPERIMENTS

This section compares the performance of ONS-OCD with COPRA [19], LFM [16], CFinder (The implementation version of CPM algorithm) [33] and OCDBIDC [30], where COPRA, LFM and CFinder are representative algorithms which detect overlapping communities directly and they are all widely accepted. So we compare these algorithms with the algorithm proposed in this paper. OCDBIDC and ONS-OCD belong to the same kind of algorithms which detect overlapping communities based on the disjoint community structure. We design this group contrast experiment to verify the performance of the proposed algorithm in this kind of algorithm.

All the simulations are carried out in a desktop PC with Intel® Core^{TM} i5-2400 3.1 GHz processor and 4 GB memory under Windows 7 OS. We implement LFM, ONS-OCD and OCDBIDC in Microsoft Visual Studio 2010 environment using C++. Other algorithms are realized with Java language.

4.1 Experimental Data

1) **LFR Benchmark Networks.** LFR benchmark networks [34, 35] are currently the most commonly used synthetic networks in community detection, including the following parameters. N is the number of nodes; $avgk$ is the average degree of nodes in the network; $maxk$ is the maximum degree of nodes; $minc$ is the number of nodes that the minimum community contains; $maxc$ is the number of nodes that the biggest community contains; mu is a mixed parameter, which is the probability of nodes connected with nodes of external community. The greater mu is, the more difficult it is to detect the community structure; om is the number of memberships of the overlapping nodes and on represents the number of overlapping nodes. We can generate different types of networks by setting different values of these parameters.

2) **Real Networks.** We also make experiments on eight well known real networks, including Zachary's karate club networks (Karate), Dolphins social networks (Dolphins), American political books networks (Polbooks), American College Football networks (Football), and so on. The detailed information of each network is shown in Table 1.

Network ID	Network Name	Number of Nodes	Number of Edges	References
R1	Karate	34	78	[36]
R2	Dolphins	62	159	[36]
R3	Political Books	105	441	[36]
R4	Football	115	613	[36]
R5	Email	1 133	5 451	[37]
R6	Political Blogs	1 490	19 090	[36]
R7	Netscience	1 589	2 742	[38]
R8	PGP	10 680	24 316	[37]

Table 1. The information of real networks

4.2 Evaluation Criteria

1) **Normalized Mutual Information (NMI).** For LFR benchmark network, we use normalized mutual information (NMI) [18] as the evaluation criteria to compare results of different algorithms, since the groundtruth of the community structure has already been known.

   Assuming the true community collection of the network is C, the membership of node i can be considered as a binary array of $|C|$ entries. If node i is present in the k^{th} community, $(x_i)_k = 1$, otherwise $(x_i)_k = 0$. We can regard the k^{th} entry of this array as the realization of a random variable X_k, whose probability distribution is $P(X_k = 1) = N_k/N$, $P(X_k = 0) = 1 - N_k/N$, where N_k is the number of nodes in the k^{th} community and N is the number of nodes in the networks. The same holds for random variable Y_l associated to the l^{th} community of the community detection result C'. We can define the conditional entropy to infer X_k given a certain Y_l, $H(X_k|Y_l) = H(X_k, Y_l) - H(Y_l)$. In particular, we can define the conditional entropy of X_k with respect to all the components of Y.

$$H(X_k|Y) = \min_{l\in\{1,2,\dots,|C'|\}} H(X_k|Y_l). \tag{5}$$

   The definition of the normalized conditional entropy of X with respect to Y is in Equation (6).

$$H(X|Y) = \frac{1}{|C|}\sum_k \frac{H(X_k|Y)}{H(X_k)}. \tag{6}$$

   The expression for $H(Y|X)$ can be determined in the same way. So, the normalized mutual information (NMI) is finally defined as Equation (7).

$$NMI(X|Y) = 1 - [H(X|Y) + H(Y|X)]/2. \tag{7}$$

   The large NMI value indicates that the community detection result is good, and vice versa.

2) **F-Measure.** For overlapping community detection algorithms, the ability of identifying overlapping nodes in the network is an important aspect to measure the performance of these algorithms. F-Measure [21] is one of the most important criteria which are widely used to measure the accuracy of algorithms in the field of machine learning. So we use F-Measure to compare the overlapping nodes detecting ability of ONS-OCD and OCDBIDC. The calculation formula of F-Measure is shown as Equation (8).

$$\text{F-Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8}$$

   where Precision indicates the ratio of the correct number of detected overlapping nodes to the total number of detected overlapping nodes (on^d). Recall is calculated by dividing the correct number of detected overlapping nodes by the total number of overlapping nodes (*on*) in the network.

The larger F-Measure value is the better the detected overlapping nodes are [39].

3) **Overlapping modularity.** As the true community structure of most real networks is unknown, we use the overlapping modularity (EQ) [7] as the evaluation criteria. It is calculated by Equation (9).

$$EQ = \frac{1}{2m}\sum_{i=1}^{K}\sum_{u,v\in C_i}\frac{1}{O_vO_u}\left(A_{uv}-\frac{d_ud_v}{2m}\right) \tag{9}$$

where m represents the number of edges in the network; A is the adjacency matrix of the network; if node u and node v are directly connected, $A_{uv} = 1$, otherwise, $A_{uv} = 0$; d_u and d_v respectively denote the degree of node u and node v. O_u and O_v respectively denote the number of communities which node u and node v belong to.

The larger EQ value is the better the result of community detection is [40].

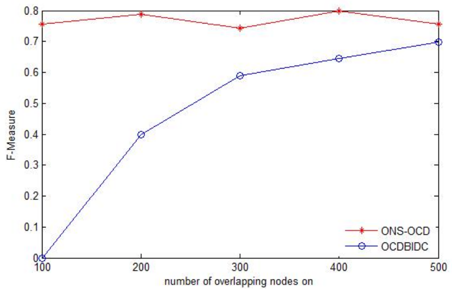
4.3 Experimental Comparison on Synthetic Networks

We use four groups of synthetic networks to evaluate the effectiveness of ONS-OCD. The details of these networks are shown in Table 2. All the networks share the common parameters of $N = 1\,000$, *avgk* = 15, *maxk* = 50 and *om* = 2. Each group contains six networks with *on* ranging from 0 to 500 and they also share parameters *minc*, *maxc* and *mu*. The community size *minc*, *maxc* are set to 10, 50 and 20, 100, respectively, implying small community networks and large community networks; *mu* is set to 0.1 and 0.3, respectively representing low and high hybrid network.

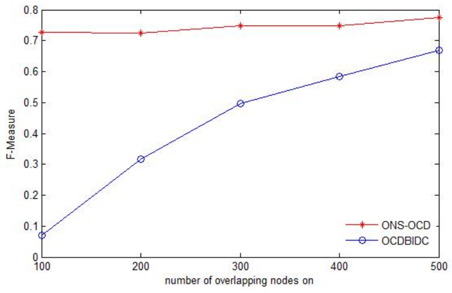
Network ID	N	*avgk*	*maxk*	*minc*	*maxc*	*mu*	*om*	*on*
S1	1 000	15	50	10	50	0.1	5	0-500
S2	1 000	15	50	10	50	0.3	5	0-500
S3	1 000	15	50	20	100	0.1	5	0-500
S4	1 000	15	50	20	100	0.3	5	0-500

Table 2. The information of four groups of LFR networks

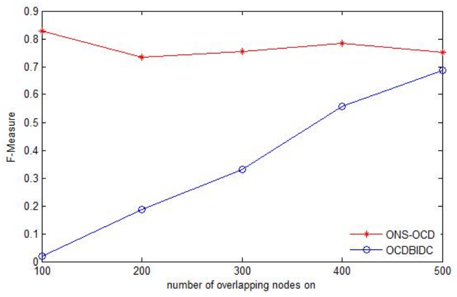
1) **The Comparison of Overlapping Nodes Detection.** First, in the case of the ideal high quality input, we compare the overlapping nodes detection ability of ONS-OCD and OCDBIDC. We do experiments on the four groups of LFR networks ($S1 \sim S4$) and choose one of the real labels of all the nodes as the input. Figure 2 depicts the results of ONS-OCD and OCDBIDC on four groups of LFR benchmark networks. The abscissa represents the number of overlapping nodes from 100 to 500, and the ordinate is the F-Measure of the results.



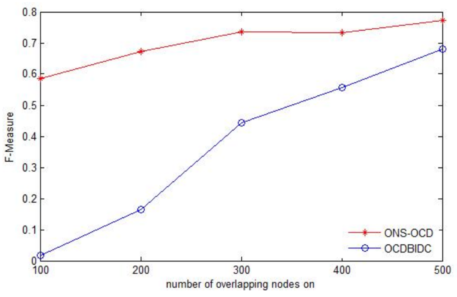
a) The F-Measure result on S1



b) The F-Measure result on S2



c) The F-Measure result on S3



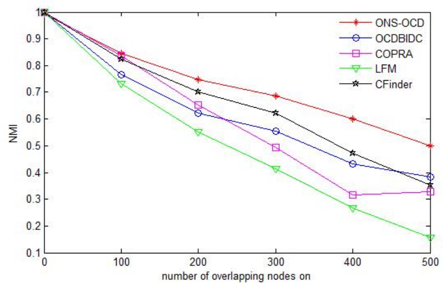
d) The F-Measure result on S4

Figure 2. The overlapping nodes detecting results of the two algorithms on LFR benchmark networks

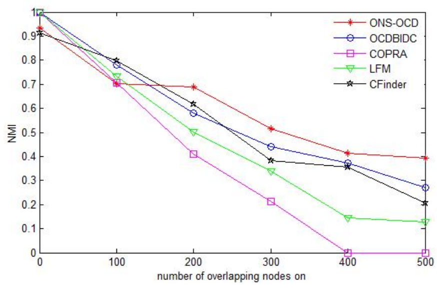
From Figure 2, it is observed that the results of the proposed algorithm are better than OCDBIDC in all these four group networks. And in the networks with different number of overlapping nodes, the overlapping nodes selection ability of ONS-OCD is basically unchanged. The overlapping nodes in the network can be well detected by ONS-OCD. In the contrast, OCDBIDC has very poor ability to detect overlapping nodes in the network with small number of overlapping nodes. When there are 100 overlapping nodes in the network, the F-Measure value of the result detected by OCDBIDC is less than 0.1. The ability of OCDBIDC to detect the overlapping nodes improves with the increase of the number of overlapping nodes in the network, but it is still worse than ONS-OCD.

2) **The Comparison of Overlapping Community Detection.** Label propagation algorithm (LPA) [12] is one of the fastest community detection algorithms, with nearly linear time complexity. The algorithm is simple and does not need any parameter, thus receiving quite a lot of attention from numerous scholars. So we use the community detection result of LPA as the input information to ONS-OCD and OCDBIDC. Three classical overlapping community detection algorithms (COPRA, LFM and CFinder) are added in this comparison. The parameters of the algorithms are set as follows: in COPRA v is varied from 2

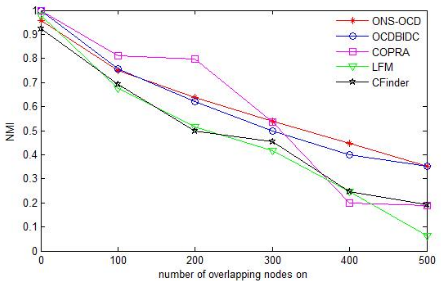
to 10 with a step size of 1; in LFM is set from 0.8 to 1.6 with a step size of 0.1; the parameter k in CFinder is initially set to 3 and increased by a step size of 1 up to 8; the parameter r of OCDBIDC is ranging from 0.1 to 1 and increased by a step size of 0.1. Each algorithm obtains different results under different parameters, and the best results of NMI are selected as the final result.



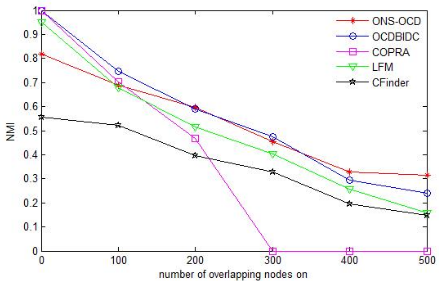
a) The NMI result on S1



b) The NMI result on S2



c) The NMI result on S3



d) The NMI result on S4

Figure 3. The overlapping community detecting results of the five algorithms on LFR benchmark networks

From these four group experimental results in Figure 3 it can be seen that in most cases, the overlapping community detection results obtained by the algorithm proposed in this paper are similar to other traditional overlapping community detection algorithms. The NMI of experimental results of all five algorithms on these four group networks decreases with the increasing number of overlapping nodes. Some traditional overlapping community detection algorithms failed to detect the overlapping community structure of the networks with too many overlapping nodes. Such as the NMI of COPRA in the second and the fourth group of networks is zero when on is larger than 400 and 300, respectively, while the results obtained by the proposed algorithm are optimal in most of these networks. From the experimental results on the third group of networks, it can be seen that COPRA is better than ONS-OCD and OCDBIDC when the number of overlapping nodes is less than 300. This is because the results of LPA on these networks are not satisfactory, which shows that the initial input of disjoint community structure has great influence on these two algorithms. In summary, the experimental results on these four groups of networks show that ONS-OCD can get good results of overlapping community structure in most cases, but it is affected by the initial disjoint community input.

4.4 Experimental Comparison on Real Networks

We still use the result of LPA as the input of ONS-OCD and OCDBIDC. The parameters of the algorithms are set as follows: in COPRA v is varied from 2 to 10 with a step size of 1; in LFM is set from 0.8 to 1.6 with a step size of 0.1; the parameter k in CFinder is initially set to 3 and increased by a step size of 1 up to 8; the parameter r of OCDBIDC is ranging from 0.1 to 1 and increased by a step size of 0.1. For the five algorithms, the maximum EQ from each result under different parameters is selected as the final result. Table 3 shows the experimental results on the eight real networks, and for every instance, the best EQ and efficiency are presented in boldface.

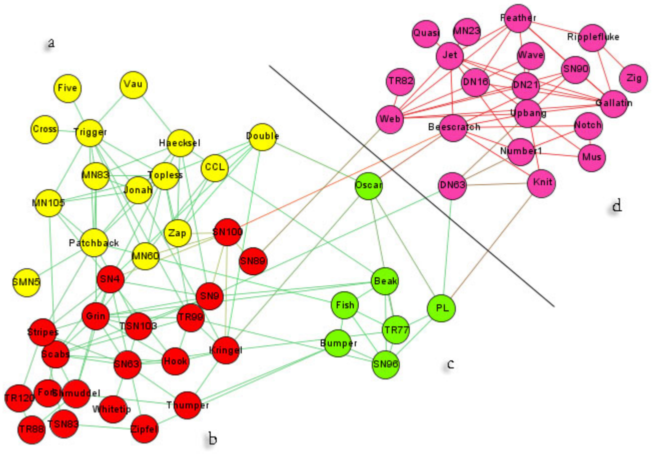
Network ID	EQ				
	COPRA	LFM	CFinder	ONS-OCD	OCDBIDC
R1	0.370	0.374	0.186	**0.733**	0.581
R2	0.204	0.436	0.361	0.730	**0.732**
R3	0.444	0.494	0.437	**0.826**	0.821
R4	0.583	0.566	0.548	**0.633**	0.620
R5	0.519	0.309	0.265	**0.650**	0.632
R6	0.765	0.748	0.758	**0.809**	0.804
R7	0.426	0.188	–	**0.913**	0.905
R8	0.780	0.622	0.389	0.811	**0.818**

Table 3. The comparison of results on real networks

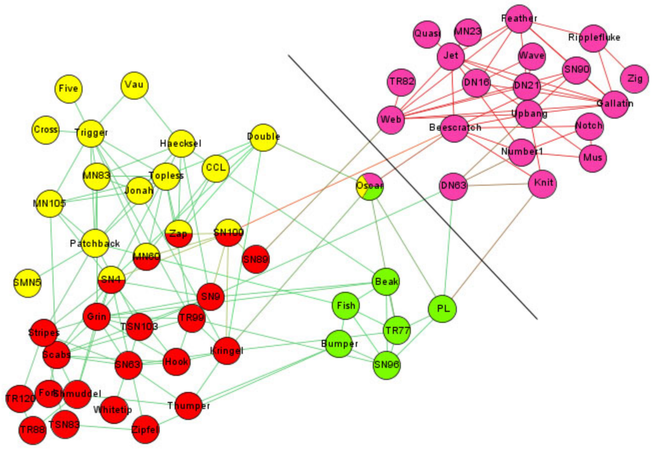
It can be seen from Table 3 that in the all real networks besides R2 (Dolphins) and R8 (PGP), the overlapping modularity of ONS-OCD is higher than those of the other four algorithms. The results of ONS-OCD on R2 (Dolphins) and R8 (PGP) are only second to OCDBIDC algorithm. Overall, the quality of the overlapping communities detected by ONS-OCD on the real networks is superior to several other algorithms.

4.5 Instance Analysis

The nodes of Dolphins are divided into two regions by a straight line in Figure 4, which represents the real division of the network. Figure 4 a) shows the community structure of Dolphins detected by LPA and Figure 4 b) is the overlapping community detection result of ONS-OCD on Dolphins. LPA algorithm divides the Dolphins data set into four communities (marked as community a, b, c and d) with different colors. It divides one of the real Dolphins communities into three. In Figure 4 b), five



a) The result of LPA on Dolphins



b) The overlapping community detection result of ONS-OCD

Figure 4. The community detection result on Dolphins

overlapping nodes (SN4, MN60, SN100, Zap and Oscar) with two or three kinds of colors are found on the basis of the result in Figure 4 a) by ONS-OCD. As can be seen in Figure 4, these nodes are closely connected between two or three communities. Node SN4 has seven adjacent nodes in the community *b*, which is obviously more than that in the community *a* which SN4 belongs to in Figure 4 a). So ONS-OCD identifies SN4 as the overlapping nodes. In summary, we can see that ONS-OCD can well identify overlapping nodes closely connected between communities.

5 CONCLUSION

In this paper, we have presented a novel overlapping node selection method to extend disjoint community structure to overlapping communities (ONS-OCD). This algorithm takes the high quality disjoint community structure as the input. Firstly, it uses the node similarity based on the heuristic DFS encoding to get the potential members of each community. Then the potential members of every community are analyzed, and the influence of the nodes on the community is calculated. Finally, the final overlapping nodes are obtained based on the node influence on communities. Since it does not need to analyze all the nodes in the network and further reduces the detection scope of overlapping nodes by the selection of potential members, it can improve the efficiency of the algorithm.

Through experiments on various synthetic networks and real networks, ONS-OCD is compared with three representative overlapping community detection algorithms (COPRA, CFinder and LFM) and OCDBIDC which also detects overlapping communities based on disjoint community structure. The results show that ONS-OCD has some advantages in the quality of community detection on the synthetic networks and real networks. In summary, ONS-OCD can identify overlapping nodes very well to get the high quality of the overlapping community structure.

Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities of Civil Aviation University of China (Grant No. 3122018C020 and 3122018C021), the Scientific Research Foundation of Civil Aviation University of China (Grant No. 600/600001050115 and 600/600001050117), the National Natural Science Foundation of China (Grant No. 61572505 and 61876186).

REFERENCES

[1] Yang, B.—Liu, D.-Y.—Liu, J.-M.—Jin, D.—Ma, H.-B.: Complex Network Clustering Algorithms. Ruan Jian Xue Bao/Journal of Software, Vol. 20, 2009, No. 1, pp. 54–66, doi: 10.3724/SP.J.1001.2009.00054.

[2] Watts, D. J.—Strogatz, S. H.: Collective Dynamics of 'Small-World' Networks. Nature, Vol. 393, 1998, No. 6638, pp. 440–442, doi: 10.1038/30918.

[3] Barabási, A.-L.—Albert, R.: Emergence of Scaling in Random Networks. Science, Vol. 286, 1999, No. 5439, pp. 509–512, doi: 10.1126/science.286.5439.509.

[4] Newman, M.—Barabási, A.-L.—Watts D. J.: The Structure and Dynamics of Networks. Princeton University Press, 2006.

[5] Newman, M. E. J.—Girvan, M.: Finding and Evaluating Community Structure in Networks. Physical Review E, Vol. 69, 2004, No. 2, Art. No. 26113, doi: 10.1103/PhysRevE.69.026113.

[6] Lee, J.—Gross, S. P.—Lee, J.: Modularity Optimization by Conformational Space Annealing. Physical Review E, Vol. 85, 2012, No. 5, Art. No. 056702, doi: 10.1103/PhysRevE.85.056702.

[7] Shen, H.—Cheng, X.—Cai, K.—Hu, M.-B.: Detect Overlapping and Hierarchical Community Structure in Networks. Physica A: Statistical Mechanics and Its Applications, Vol. 388, 2009, No. 8, pp. 1706–1712, doi: 10.1016/j.physa.2008.12.021.

[8] Shen, H.-W.—Cheng, X.-Q.: Spectral Methods for the Detection of Network Community Structure: A Comparative Analysis. Journal of Statistical Mechanics: Theory and Experiment, Vol. 2010, 2010, No. 10, Art. No. P10020, doi: 10.1088/1742-5468/2010/10/P10020.

[9] Huang, L.—Li, R.—Chen, H.—Gu, X.—Wen, K.—Li, Y.: Detecting Network Communities Using Regularized Spectral Clustering Algorithm. Artificial Intelligence Review, Vol. 41, 2014, No. 4, pp. 579–594, doi: 10.1007/s10462-012-9325-3.

[10] Girvan, M.—Newman, M. E. J.: Community Structure in Social and Biological Networks. Proceedings of the National Academy of Sciences of the United States of America (PNAS), Vol. 99, 2002, No. 12, pp. 7821–7826, doi: 10.1073/pnas.122653799.

[11] Blondel, V. D.—Guillaume, J.-L.—Lambiotte, R.—Lefebvre, E.: Fast Unfolding of Communities in Large Networks. Journal of Statistical Mechanics: Theory and Experiment, Vol. 2008, 2008, No. 10, Art. No. P10008, doi: 10.1088/1742-5468/2008/10/P10008.

[12] Raghavan, U. N.—Albert, R.—Kumara, S.: Near Linear Time Algorithm to Detect Community Structures in Large-Scale Networks. Physical Review E, Vol. 76, 2007, No. 3, Art. No. 036106, doi: 10.1103/PhysRevE.76.036106.

[13] Šubelj, L.—Bajec, M.: Unfolding Communities in Large Complex Networks: Combining Defensive and Offensive Label Propagation for Core Extraction. Physical Review E, Vol. 83, 2011, No. 3, Art. No. 036103, doi: 10.1103/PhysRevE.83.036103.

[14] Rosvall, M.—Bergstrom, C. T.: Maps of Random Walks on Complex Networks Reveal Community Structure. Proceedings of the National Academy of Sciences of the United States of America (PNAS), Vol. 105, 2008, No. 4, pp. 1118–1123, doi: 10.1073/pnas.0706851105.

[15] Palla, G.—Derényi, I.—Farkas, I.—Vicsek, T.: Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society. Nature, Vol. 435, 2005, No. 7043, pp. 814–818, doi: 10.1038/nature03607.

[16] Lancichinetti, A.—Fortunato, S.—Kertész, J.: Detecting the Overlapping and Hierarchical Community Structure in Complex Networks. New Journal of Physics, Vol. 11, 2009, No. 3, Art. No. 033015, doi: 10.1088/1367-2630/11/3/033015.

[17] Lancichinetti, A.—Radicchi, F.—Ramasco, J. J.—Fortunato, S.: Finding Statistically Significant Communities in Networks. PLoS ONE, Vol. 6, 2011, No. 4, Art. No. e18961, doi: 10.1371/journal.pone.0018961.

[18] Coscia, M.—Rossetti, G.—Giannotti, F.—Pedreschi, D.: DEMON: A Local-First Discovery Method for Overlapping Communities. Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12), Beijing, 2012, pp. 615–623, doi: 10.1145/2339530.2339630.

[19] Gregory, S.: Finding Overlapping Communities in Networks by Label Propagation. New Journal of Physics, Vol. 12, 2010, No. 10, Art. No. 103018, doi: 10.1088/1367-2630/12/10/103018.

[20] Wu, Z.-H.—Lin, Y.-F.—Gregory, S.—Wan, H.-Y.—Tian, S.-F.: Balanced Multi-Label Propagation for Overlapping Community Detection in Social Networks. Journal of Computer Science and Technology, Vol. 27, 2012, No. 3, pp. 468–479, doi: 10.1007/s11390-012-1236-x.

[21] Xie, J.—Szymanski, B. K.—Liu, X.: SLPA: Uncovering Overlapping Communities in Social Networks via a Speaker-Listener Interaction Dynamic Process. Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops (ICDMW '11), Vancouver, 2011, pp. 344–349, doi: 10.1109/ICDMW.2011.154.

[22] Ahn, Y.-Y.—Bagrow, J. P.—Lehmann, S.: Link Communities Reveal Multiscale Complexity in Networks. Nature, Vol. 466, 2010, No. 7307, pp. 761–764, doi: 10.1038/nature09182.

[23] Kim, Y.—Jeong, H.: Map Equation for Link Communities. Physical Review E, Vol. 84, 2011, No. 2, Art. No. 026110, doi: 10.1103/PhysRevE.84.026110.

[24] Huang, F.-L.—Xiao, N.-F.: Discovering Overlapping Communities Based on Line Graph and PSO. Zidonghua Xuebao/Acta Automatica Sinica, Vol. 37, 2011, No. 9, pp. 1140–1144.

[25] Pan, L.—Jin, J.—Wang, C.-J.—Xie, J.-Y.: Detecting Link Communities Based on Local Information in Social Networks. Tien Tzu Hsueh Pao/Acta Electronica Sinica, Vol. 40, 2012, No. 11, pp. 2255–2263.

[26] Yu, L.—Wu, B.—Wang, B.: LBLP: Link-Clustering-Based Approach for Overlapping Community Detection. Tsinghua Science and Technology, Vol. 18, 2013, No. 4, pp. 387–397, doi: 10.1109/TST.2013.6574677.

[27] Shi, C.—Cai, Y.—Fu, D.—Dong, Y.—Wu, B.: A Link Clustering Based Overlapping Community Detection Algorithm. Data and Knowledge Engineering, Vol. 87, 2013, pp. 394–404, doi: 10.1016/j.datak.2013.05.004.

[28] Chakraborty, T.: Leveraging Disjoint Communities for Detecting Overlapping Community Structure. Journal of Statistical Mechanics: Theory and Experiment, Vol. 2015, 2015, Art. No. P05017, doi: 10.1088/1742-5468/2015/05/P05017.

[29] Wang, X.—Jiao, L.—Wu, J.: Adjusting from Disjoint to Overlapping Community Detection of Complex Networks. Physica A: Statistical Mechanics and Its Applications, Vol. 388, 2009, No. 24, pp. 5045–5056, doi: 10.1016/j.physa.2009.08.032.

[30] Li, Y.—Liu, G.—Lao, S.-Y.: Overlapping Community Detection in Complex Networks Based on the Boundary Information of Disjoint Community. 2013 25th Chinese Control and Decision Conference (CCDC), Guigang, 2013, pp. 125–130, doi: 10.1109/CCDC.2013.6560906.

[31] Tang, L.—Liu, H.: Community Detection and Mining in Social Media. Morgan and Claypool Publishers, 2010, doi: 10.2200/S00298ED1V01Y201009DMK003.

[32] Yu, T.—Xiao, Y. H.—He, Z. Y.—Wu, W. T.: Fast Randomized Algorithm for Community Detection in Large Networks. Journal of Computer Research and Development, Vol. 46, 2009, pp. 406–412.

[33] `http://cfinder.org/`.

[34] Lancichinetti, A.—Fourtunato, S.—Radicchi, F.: Benchmark Graphs for Testing Community Detection Algorithms. Physical Review E, Vol. 78, 2008, No. 4, Art. No. 046110, doi: 10.1103/PhysRevE.78.046110.

[35] Lancichinetti, A.—Fortunato, S.: Benchmarks for Testing Community Detection Algorithms on Directed and Weighted Graphs with Overlapping Communities. Physical Review E, Vol. 80, 2009, No. 1, Art. No. 016118, doi: 10.1103/PhysRevE.80.016118.

[36] `http://www-personal.umich.edu/~mejn/netdata/`.

[37] `http://www.cs.bris.ac.uk/~steve/networks/copra/`.

[38] Newman, M. E. J.: Finding Community Structure in Networks Using the Eigenvectors of Matrices. Physical Review E, Vol. 74, 2006, No. 3, Art. No. 036104, doi: 10.1103/PhysRevE.74.036104.

[39] Zhou, X.—Liu, Y.—Zhang, J.—Liu, T.—Zhang, D.: An Ant Colony Based Algorithm for Overlapping Community Detection in Complex Networks. Physica A: Statistical Mechanics and Its Applications, Vol. 427, 2015, pp. 289–301, doi: 10.1016/j.physa.2015.02.020.

[40] Zhou, X.—Liu, Y.—Wang, J.—Li, C.: A Density Based Link Clustering Algorithm for Overlapping Community Detection in Networks. Physica A: Statistical Mechanics and Its Applications, Vol. 486, 2017, pp. 65–78, doi: 10.1016/j.physa.2017.05.032.

Yan Xing is currently Lecturer in School of Computer Science and Technology, Civil Aviation University of China. Her research interests include data mining, complex network and community detection.

Fanrong Meng is Professor at the School of Computer Science and Technology, China University of Mining and Technology. Her research interests include database technology, data mining and knowledge discovery.

Yong Zhou is Professor at the School of Computer Science and Technology, China University of Mining and Technology. His research interests include data mining, genetic algorithm, artificial intelligence and wireless sensor network.

Guibin Sun is a master student at the School of Computer Science and Technology, China University of Mining and Technology. His research interests include data mining, complex network and community detection.

Zhixiao Wang is Professor at the School of Computer Science and Technology, China University of Mining Technology. His research interests include field theory application, and social network analysis.

Computing and Informatics, Vol. 38, 2019, 1111–1130, doi: 10.31577/cai_2019_5_1111

INFORMATION TECHNOLOGY OF GENERALIZED MODEL CREATION OF COMPLEX TECHNICAL OBJECTS

Dmytro Konotop, Valeriy Zinchenko

National Technical University of Ukraine
Kyiv Polytechnic Institute
Peremogy ave 37
03056 Kyiv, Ukraine
e-mail: konotop.dmitriy@gmail.com

Ivana Budinská

Institute of Informatics
Slovak Academy of Sciences
Dúbravská cesta 9
845 07 Bratislava, Slovakia
e-mail: budinska@savba.sk

Wei Li

Shenyang Aerospace University
No. 37 Daoyi South Avenue
Shenbei New Area
Shenyang, China 110136

Abstract. The paper introduces a knowledge representation framework for design and geometrical modelling of complex technical objects such as ships, aircrafts, cars, etc. The design process cannot be fully automated yet because of a lot of technical and economical factors that influence the decisions during that process. In order to make the process more efficient, a knowledge modelling framework is suggested. The basic principles of conceptual knowledge modelling and data exchange framework are presented. A practical use case of aircraft ramp modelling is provided.

Keywords: Knowledge base, geometric model, managing parametric model

1 INTRODUCTION

Aircraft creation process is very complicated and important. Now the aircraft models are almost not linked and the geometric models connections between different design stages are also almost absent. Knowledge-based approach of aircraft geometrical modelling is trying to decide these problems. Tasks and features of the knowledge-based systems adoption in aircraft geometric models creation are described. In this article a knowledge-based approach of aircraft geometrical modelling is represented. Also the knowledge-based approach modification for parametric information control of aircraft geometrical modelling and the modification for data transferring of aircraft geometrical modelling are shown. Realization of the knowledge-based approach of aircraft geometrical modelling and its estimation are made. The task of any complex technical object creation (CTO), which is characterized by a large quantity of elements and different links, is the development of chart, structure and construction of the future CTO and constituents of its elements, which must provide at certain limitations the most effective performance of the put goals. Fully automated process of CTO creation is impossible, because of a lot of factors, as technical and economic, influence on decisions, which are adopted in this process. The main task of information technology adoption in the process of CTO creation is the best possible simplification of operations by automation of the most possible number of CTO creation tasks. This task is resolved by the development of generalized model of CTO and knowledge-based modelling approach of CTO.

1.1 The Main Problems of Modern Complex Technical Object Creation Process

Modern complex technical objects, which are characterized by a large quantity of elements and different links (ships, aircrafts, space techniques, cars, etc.), are created using such information technology (IT) as continuous acquisition and life cycle support (CALS), which includes the main information systems (IS): product lifecycle management (PLM) and computer-aided systems (CAx). Different models of CTO describe the relationships between CTO parameters and its characteristics [1]. Modern CTO creation includes following main stages: requirements specification (RS) and draft proposal (DP), master-geometry model (MGM, conceptual design), objects allocation model (OAM, preliminary design) and complete product definition model (CPDM, detail design) [2, 3]. Compliance of the classic CTO creation process and modern CTO modelling using IT are represented in Figure 1. Over

75 % of the basic technical and organizational solutions for the project are taken at the conceptual and preliminary design stages at costs up to 20 % of time and 10 % of the funds [1, 3, 4].

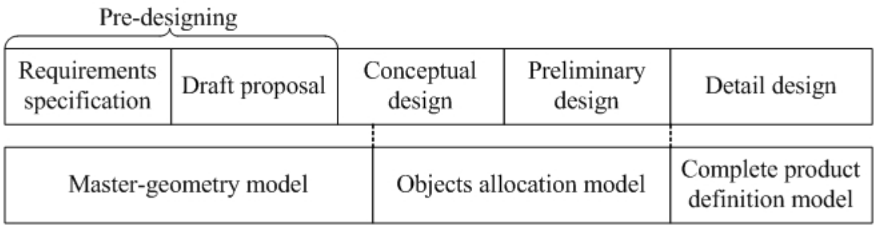


Figure 1. Compliance of the classic CTO creation and modern modelling

The level of structural synthesis problem complexity applied to CTO creation is very difficult. Effective solution of these problems is only possible when using CALS. However, CALS usage focuses on formalized statement of the structural synthesis problem. The basis of such formalized statement is the formulation of structural synthesis problem as a mathematical problem of design decisions making (DDM) as follows [5]:

$$DDM = \langle A, C, M, R \rangle$$

where

- A – alternatives set of design making;
- $C = (C1, C2, \ldots, C)$ – criteria set (initial parameters), which evaluated alternative compliance goals;
- M – model, which allows for each alternative to calculate the criteria vector;
- R – decision rule for selecting a suitable alternative to multi situations.

The developed process of CTO creation can be represented as the following generalized iterative procedure:

$$\begin{tikzcd}
{} \arrow[r] & A_0^j \arrow[r] & M_1^i \arrow[r] & \langle R_1 \rangle \arrow[r] \arrow[l, bend left, "(i+1)"] & M_2^i \arrow[r] & \langle R_2 \rangle \arrow[r] \arrow[l, bend left, "(i+1)"] \arrow[lll, bend right] & M_3^i \arrow[r] & \langle R_3 \rangle \arrow[r] \arrow[l, bend left, "(i+1)"] \arrow[lll, bend right] & \langle R_0(\mathrm{T3}) \rangle \arrow[r] \arrow[lllllll, bend left, "(j+1)"] & {}
\end{tikzcd}$$

where:

- A_0 – initial design data (pre-designing),
- M_i – models: MGM(M_1), OAM(M_2) and CPDM(M_3),
- $(i + 1)$ – new (changed) parameters, design updating, appearing on the results of the next design stages, can then be modified in the previous stages.

Any model on different stages is possible to present as:

$$M_i = \sum_{i=1}^{N} m_i$$

where:

- m_i are the components of the appropriate design stage,
- N are the models set.

The main problems of modern complex technical object creation process are:

1. CTO models include parts and assembly units created by CAx and PLM systems and often contain many expressions between components, references, restrictions, etc., and that requires a detailed description of the process of CTO creation.
2. CTO models at various stages of CTO creation are not linked. That is, if there are changes in the components of models at the early design stages, these changes do not appear in the detail design stage of CTO creation.
3. Described various CTO models are actually unrelated in the process of CTO creation with the use of CAx and PLM systems. Many data in general remain beyond a joint project of CTO development.
4. CTO creation involves many disciplines and communication between different units. As for design techniques and tools, it is desirable that the design takes place in parallel with further manufacturing and assembly, which is currently extremely difficult, exhausting and almost unrelated procedure due to weak production automation that prevents the use of the process of CTO models.
5. CTO models creation is also using different systems CAx and PLM, which creates a constant difficulty of data converting from one software to another and leads to a partial or complete loss of generalized model components, its history, topology and parameters. To reduce the time-to-market and to minimize errors when displaying CTO models, new approaches must be applied in data transferring between different systems CAx and PLM. Delays in setting up controversial issues when exchanging data will cause many recycles in CTO creation and thus causing a significant delay in the project schedule.

Considering these problems, the relevant scientific and practical task of work follows – to research and develop the generalized model of CTO, which is able to solve the task of different models links on different stages of CTO creation lifecycle, and to develop the knowledge-based modelling framework of CTO based on the developed generalized model.

2 TASKS AND FEATURES OF THE KNOWLEDGE BASED SYSTEMS ADOPTION IN AIRCRAFT GEOMETRIC MODELS CREATION

The generalized model of CTO contains the main CTO models from different IS of CALS at the different CTO creation lifecycle; this is researched, developed and presented as:

$$MGM = \{MG, MW, MA, MAA, MS, MPP, MT, ME\}$$

where the following models appear: geometrical; weight; aerodynamic; arranging and alignment; strength; power plant; manufacturability; economical, respectively. Any model contains the common information with other models and owner information. Any component of the models is described by owner set of parameters and can be represented as:

$$m_i = f(P_i)$$

where P_i – models parameters: $P_i = \sum_{i=1}^{N} p_i$.

The generalized model of CTO component can be represented as:

$$m_{GM} = f^i(p_i,\ i = 1, \dots n).$$

$m_{GM} = (m_1, \dots, m_n)$ – is a vector of parts of CTO generalized model of component creation, any of them contains the defined restrictions (geometrical; weight; manufacturability, etc.) depending on parameter type, $m_i \in M_i$.

The aim of work is formulated as the definition on the permissible set of decision variants, which are described by restrictions of such variants so that the optimality criterion (objective function or functional), which defines the decision quality, accepts the extreme value:

$$F^* = extr_{P_n \in E_k(m_{GM})} F(P_n)$$

where $E_k(m_{GM})$ – parameters efficiency criteria $m_{GM} \in M_{GM}$.

Systems of restrictions looks like:

$$\left\{ \begin{array}{l} E_1(m_{GM}) \geq 0 \\ E_2(m_{GM}) \geq 0 \\ \dots \\ E_n(m_{GM}) \geq 0 \end{array} \right\}.$$

In particular, the inequality of the system $E_1(m_{GM})$ has a geometrical character; CTO creation specific puts also restrictions: $E_2(m_{GM})$ – alignment; $E_3(m_{GM})$ – CTO weight; $E_4(m_{GM})$ – by external influences on the development area (mechanical, climate, etc.), equipment compatibility; $E_5(m_{GM})$ – implementation of manufacturability requirements, which are the interlink between CTO design and manufacture, etc. Acceptable level of $E_k(m_{GM})$ makes mention of some normative document, CTO creation requirements, RS, etc. Knowledge-based modelling approach is a basis for IT development. Developed approach supplements and extends the current methods of CTO creation using IT and defines the main design stages of the knowledge-based system (KBS) adoption in CTO models building process. The development of engineering knowledge project management (Managing Engineering Knowledge, MOKA) had tremendous significance in the decision of knowledge representation problem in CTO creation [6]. In the project of MOKA, the unified modelling language was used for knowledge representation. The area of the knowledge using comes into question very often; some researchers assume that any object can be represented as area of knowledge. Other works conflict with this principle [7] and suggest to build small areas which are easier to support. But in any case, each researcher realizes the importance of the knowledge using in CTO creation.

Theory and practice of the creation and use of KBS is the most actual direction of computer sciences that has been intensively developing. Using the results promotes efficiency of creation of tools, application systems and computers application. At work KBS is included for combination of such main CTO creation processes (Figure 2).

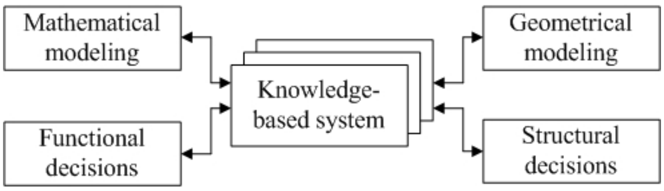


Figure 2. Place of the knowledge based system in CTO design

On the existing stage of development of CAx and PLM systems the main interest lies in the use of the knowledge in CTO creation, which would allow a subsequent perfection of the process of CTO creation, and this process can be named knowledge based modelling of CTO – it includes the use of suitable software for acquisition and reusing the knowledge in CTO creation by the most possible complex approach. The tasks which are executed by traditional methods, mainly, do not cause unexpected problems, but they are tiresome and by such approach labor intensive and expensive (from the economic point of view). According to Stokes [6], the percentage of time spent on the conservative jobs processing at CTO creation is approximately 80 %.

KBS adoption in the process of CTO GM creation is related to reusing the knowledge that comes from previous developments. Developing the approach of KBS adoption in CTO creation allows to solve the following tasks: to describe the process of CTO creation in details, to apply the modern KBS; to create the links of GM on the different stages of CTO creation; to link GM with other CTO models; to apply new approaches at data transferring between different CAx and PLM systems. For the effective solution of the listed tasks it is necessary to analyze the main features of subject domain of CTO creation and to develop a feasible approach and facilities for implementation of automation in this area [8]. Extended tasks of KBS adoption in the process of CTO creation are shown in Figure 3.

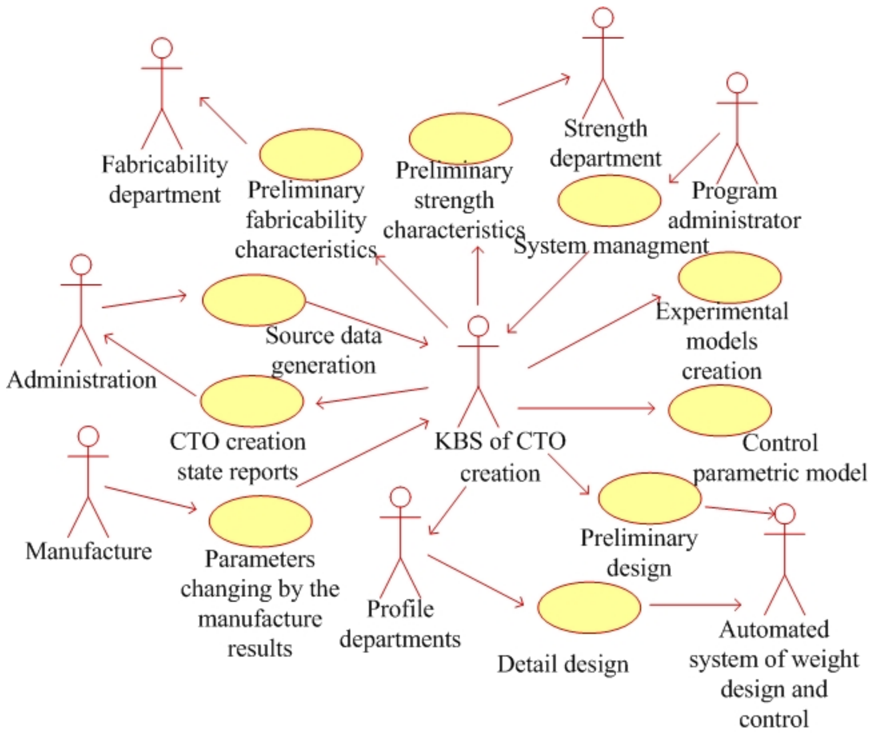

Fabricability department
Preliminary fabricability characteristics
Preliminary strength characteristics
Strength department
Program administrator
System managment
Administration
Source data generation
Experimental models creation
CTO creation state reports
KBS of CTO creation
Control parametric model
Manufacture
Parameters changing by the manufacture results
Profile departments
Preliminary design
Detail design
Automated system of weight design and control

Figure 3. Tasks of KBS adoption in the process of CTO creation

The approach of KBS adoption in CTO creation and the subsequent modification of the approach are designed – to manage parametric information; to exchange the data. Let us analyse the use of knowledge in modern systems CALS order to ensure feasibility of KBS adoption in the process of CTO creation. Consider the examples of knowledge using in the most popular systems CAx at a high level. Pro/ENGINEER or PTC Creo system from PTC Company contains elements of knowledge to use standardized conceptual design and has the libraries: library of standard two-dimensional layouts; library of standard three-dimensional layouts; library of standard drafts, separate components, standard design solutions [9]. CATIA, ENOVIA and DELMIA systems from the Dassault Systèmes Company allow to maintain the enterprise rules-based design and knowledge reuse. Controlled methodology that uses skeletal geometry allows a quick changing of specifications and knowledge reuse [10]. The module "Knowledgeware" has the following most important sub-modules: knowledge adviser; knowledge expert; knowledge of product template; business process knowledge template. NX system from Siemens PLM Software Company is built on knowledge architecture that provides unlimited opportunities for the application in the knowledge process design and experience gained by company or industrial "know-how" [11]. The system raises the design process to another level with the DesignLogic technology, which manages the product using the embodied knowledge in the form of functions and formulas, associative dimensions and links. All the above listed elements of knowledge using contained in various CAx systems have significant disadvantages – for instance the knowledge used in the existing CAx attached directly to the system makes it impossible to use the universal knowledge base in different CAx.

3 KNOWLEDGE-BASED MODELLING FRAMEWORK

3.1 Conceptual Modelling

Considering tasks which stand before the developer of methods and facilities of KBS adoption in the process of CTO creation, it is possible to define the features of their structure and functioning. The analysis of subject domain is the special type of scientific activity. The result of analysis of a subject domain is built of interpretation model of the subject knowledge [13]. CTO creation efficiency is largely conditioned using system approach like its constituents, and as systems engineering and systems analysis (SA). Methods of SA are the basis of CASE-technology (Computer-Aided Software Engineering, automation of processes of planning and software development). Methods of SA are described by the series of IDEF standards of ICAM (Integrated Computer-Aided Manufacturing) DEFinition [IDEF], which are used for task decision making on the design of difficult systems. For example, IDEF5 is standard of KBS (ontological research of the difficult systems) [14]. By IDEF5 methodology, the ontology of the system can be described by the certain dictionary of terms and rules on the basis of which the reliable assertions about the state of this system can be formed in some moment of time. By the opinion of specialists of SA area [15, 16], for the decision of analysis tasks and planning of some object there are designed: functions of this object, for example, by the diagrams of data flows – Data Flow Diagram (DFD); relation between data which are used in object, for example, by diagrams "essence-connection" of Entity-Relations Diagram (ERD); behavior of the object (event), for example, using Activity diagrams. The functions (processes), depositories of data and streams which link them, are represented by DFD. Presentation of object functions, as a rule, occurs at a few levels of detailing, and the content of the processes shown at previous level is opened up at every next level. In Figure 4, the context DFD of KBS adoption in CTO creation is represented.

The built DFD allows defining the basic stages of work on KBS adoption in the process of CTO creation, and also basic types of data sets with which work

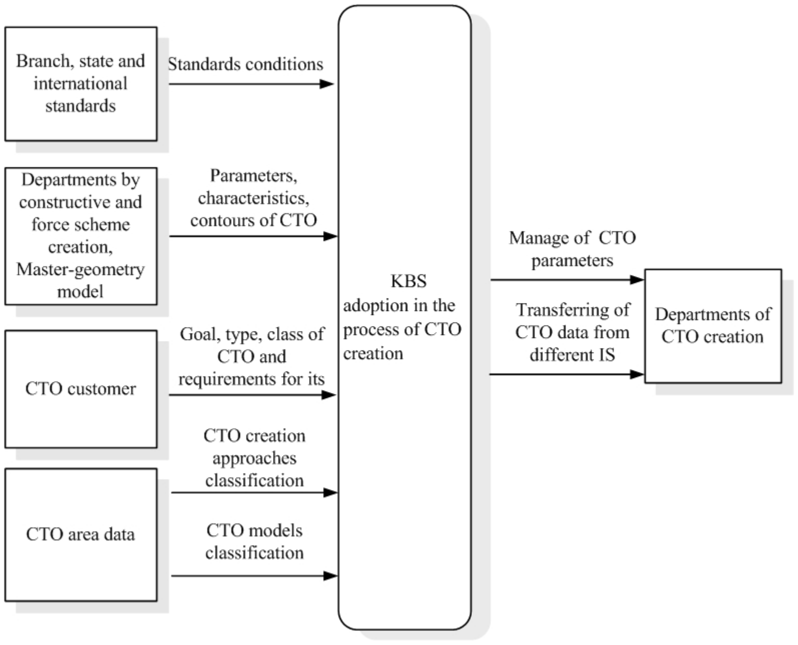

Branch, state and international standards
Standards conditions
Departments by constructive and force scheme creation, Master-geometry model
Parameters, characteristics, contours of CTO
CTO customer
Goal, type, class of CTO and requirements for its
CTO area data
CTO creation approaches classification
CTO models classification
KBS adoption in the process of CTO creation
Manage of CTO parameters
Transferring of CTO data from different IS
Departments of CTO creation

Figure 4. The context DFD of KBS adoption in the process of CTO creation

will be in the process of functioning of the system: parameters; output data and results of GM creation; terms of classifications, correlations, rules of interpretation, norms, and requirements of standards. Entity-Relationship Diagram is intended for development of data models and provides the standard approach of determination of relations between them. These KBS of CTO creation are organized in three structures: knowledge base (KB), that contains the terms from the CTO area, connections between them, rules which establish order of the terms interpretation, application of knowledge from a knowledge base, conditions of calculations, concordance with the requirements of standards; database (DB), that contains files of CTO creation, results of calculations, different service information located in PLM system [17]; file DB contains parameters for CTO creation determination. In Figure 5, ERD of the main data sets of KBS of CTO creation is represented.

Analyzing the main tasks, requirements, features of functioning and order of implementation of operations of CTO creation, it is possible to formulate the general features of KBS adoption of CTO creation: work with large data sets; iteration character of CTO creation on the different stages of GM creation; CTO modelling realization by different methods, the choice of which depends on a type, configuration and parameters of CTO; the necessity of selection and use of separate parameters and characteristics of CTO creation which will be used from previous stages of CTO

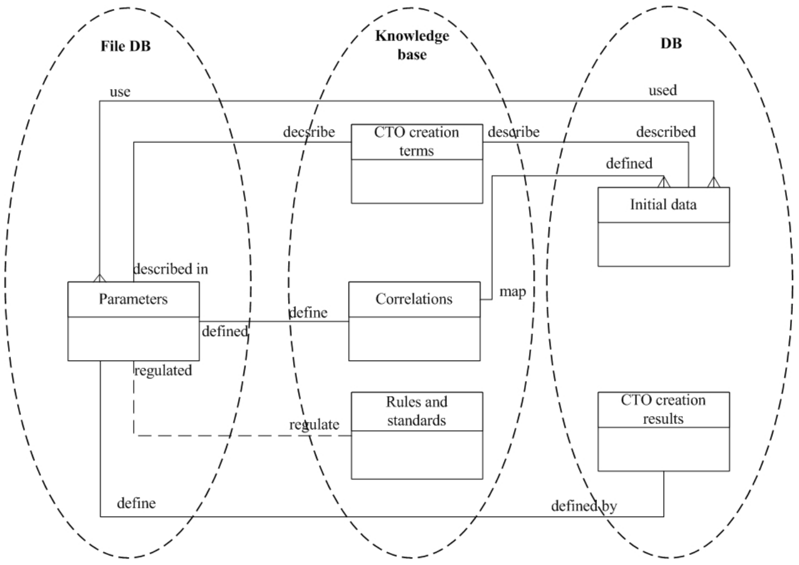


Figure 5. The context DFD of KBS adoption in the process of CTO creation

creation and finishing the CTO production; necessity of providing a connection between different CTO models.

3.2 Parametric Information Control

Control parametric model (CPM) is the aggregate of basic data which is the basis for CTO creation at any stage, and it is a hierarchical structure in CAx/PLM system. CPM is the base of technical information in an electronic view, on the basis of which development of CTO is conducted in accordance with thematic directions of the CTO project [18]. The geometry, which is passed from the previous stage of CTO creation – master-geometry model is used at CPM building. CPM is the aggregate of geometrical elements which are used in the process of GM creation. CPM differs from GM, which is the typical result of the CAx system, and the universal presentation of CTO. CPM is built by rules which determinate the project and are the basis for GM (Figure 6).

Approach modification for parametric information control (Figure 7 as the activity diagram) provides for creation of the interconnection of CTO models development stages.

3.3 Data Transferring

Disadvantages arise while creating CTO in different IS CAx. Most of them are connected with the exchange of data from different models. Further, it proposes

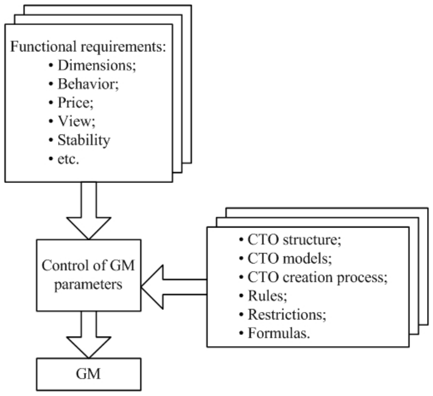


Figure 6. Control of generalized model parameters

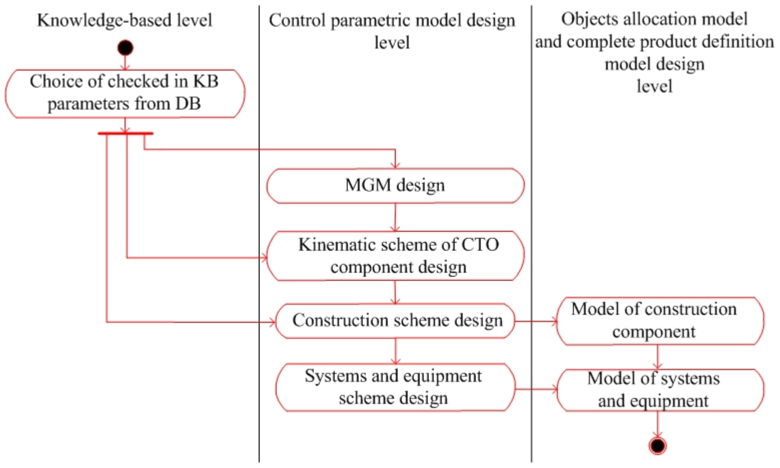

Knowledge-based level
Control parametric model design level
Objects allocation model and complete product definition model design level
Choice of checked in KB parameters from DB
MGM design
Kinematic scheme of CTO component design
Construction scheme design
Systems and equipment scheme design
Model of construction component
Model of systems and equipment

Figure 7. Approach modification for parametric information control

that a knowledge-based approach in communicating CTO models is created in different IS CAx by developing a modification of the knowledge-based modelling approach. Data exchange between CAx-systems is mainly using translators to facilitate the exchange between models from CAx-systems. Data loss during transmission leads to the fact that models transferred between different systems are incomplete. Available compilers developed for specific CAx-systems are expensive and not universal.

Neutral standards should provide a single interface to information applied software. The most used standards: Standard Generalized Markup Language; Computer Graphic Metafile; Initial Graphics Exchange Specification [19]; Standard for Exchange of Product model data [20]; STL format – Stereolithography. A software library for the programming language Javascript that allows to create Javascript interactive 3D-graphics, operating in a wide range of compatible web browsers (Mozilla Firefox, Google Chrome, Opera and Internet Explorer), supporting HTML, without the mediation of plug-ins, is distributed by the use of WebGL standard (based on OpenGL) [21]. And this standard has serious disadvantages: models are built using WebGL and are kept only in the format of STL, which is not supported by all CAx-systems; WebGL also contains many significant security problems, in particular, the arbitrary code execution and possible cross-domain attacks on models saved.

These data formats between different CAx-systems are capable of handling full set of information from different CAx-systems and transfer all the information about the models, including the history of building and constraints within it. A lot of different methods are described on how to implement data exchange between different CAx-systems, each of which has its advantages and disadvantages. General scheme of data exchange with models from different IS CAx using KBS is shown in Figure 8.

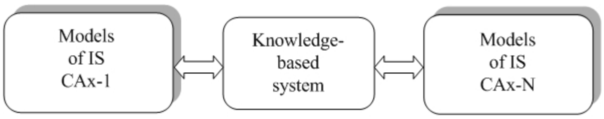


Figure 8. Data exchange with models from different IS CAx using KBS

Approach modification for data transferring from different IS CAx, using KBS is shown as the diagram of activity (Figure 9) [22].

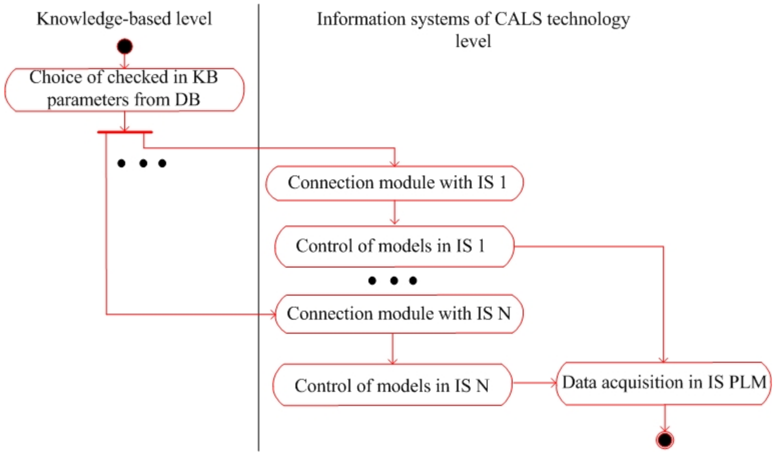

Knowledge-based level
Information systems of CALS technology level
Choice of checked in KB parameters from DB
Connection module with IS 1
Control of models in IS 1
Connection module with IS N
Control of models in IS N
Data acquisition in IS PLM

Figure 9. Approach modification for data transferring from IS CAx using KBS

4 USE CASE: KNOWLEDGE BASED ENGINEERING DESIGN

Based on the features development means of KBS, the CTO models creation is selected, with the main features of the development, and it identifies the main means of implementing the methods of implementation of KBS in CTO creation.

A variety of information which affects the decision to incorporate this information as well as to store data about the state of the system and to facilitate its management is processed. All this information should work together to create the knowledge base (KB), which includes a global information system as a whole and combines its separate parts.

The system must work with large amounts of data on GM parts and assembly units. A database should be developed for saving this information. From the set of data stored in the database, you can select entities, such as dimensions, limitations of CTO, etc. that have certain attributes and relationships with each other. Therefore, the development and implementation of the database (DB) can be made using the model "entity-relationship".

The multi-agent systems are used. Since the task and the process of CTO creation are closely linked, it is difficult to identify independently solvable subtasks that can be executed simultaneously regardless of the computer, so the use of intelligent agents in this case is not appropriate, however non-intelligent agents may be used.

Guided by the principles of client-server technology and structured approach, we logically choose the implementation of knowledge-based systems in the design of CTO structure of Web-based applications. There are distributed softwares that use the basic infrastructure of the Internet for communication between its components and standard tools navigation Web-browser – as the foundation for the user interface.

In work guided by the principles of visual programming, paradigms and tools are based on J2EE technologies.

The work shall be chosen by means of CALS technology adopted to implement the methods in the form of systems CAx and PLM.

Ontology of some industry knowledge, together with information about the properties of specific objects, can be called knowledge base (KB) [23]. The knowledge base chart of CTO creation process is shown in Figure 10.

CTO creation ontology is shown in Figure 11.

It describes the main connection and correlation between the main parts of the design process under development of CTO at the earliest design stages. DB chart "CTO models parameters" in the example of an aircraft is shown in Figure 12.

Ontology of CTO creation was built by the tools of Protègè 4.3 [28, 24], its connectivity and correctness was checked up by the instruments of this application. The functions of addition, change and verification of ontology are carried out by the application programs of this system, and also tools of the proper libraries of Java, Jena, Pellet. An ontological KB plays a key role in realization of the process of KBS

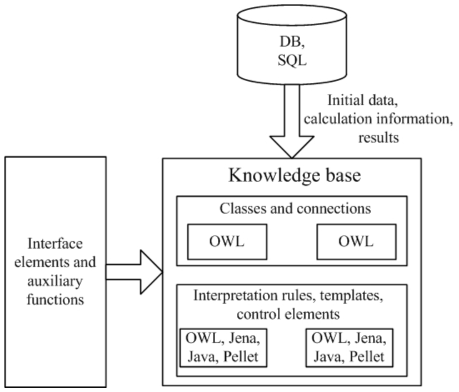

DB,
SQL
Initial data, calculation information, results
Knowledge base
Classes and connections
OWL
OWL
Interface elements and auxiliary functions
Interpretation rules, templates, control elements
OWL, Jena, Java, Pellet
OWL, Jena, Java, Pellet

Figure 10. Knowledge base chart

adoption at CTO creation process. The KBS adoption in CTO components circuit is shown in Figure 13.

The components of KBS adoption in CTO creation are:

1. Module working with KB, DB and decision making: basic module system organizes the interaction of all its parts; it supplies the conduct of process designing, based on the data from KB and DB.
2. Module working with CAx systems: it organizes iteration plans and conducts designing steps by organization of connection with CAx systems, and subsequently, with PLM.
3. Information security module: it supplies the user identification and authentication; it organizes various types of access to KBS adoption in CTO creation process.

Figure 14 presents the data exchange from CPM to GM. In particular, something with the change of the aircraft ramp in DB makes automatically the changes coming from the kinematic scheme of CPM of the aircraft ramp.

Some authors [1, 25, 26] suggest that the average duration of the aircraft creation project cycle takes 6 years. It is important to bear in mind that this does not include the development, which is given for early configuration and market analysis. The duration of modelling is an important economic characteristic of CTO creation, because it often defines the general terms of product development, and that is, the speed of the project realization. Reduction of time determined by the entire lifecycle of CTO cannot be achieved by significantly increasing the productivity specialists – programmers, designers, production engineers, etc. Also, it is less dependent on the software being used. Adoption of new approaches of work organizing the CTO creation reduces significantly time for the CTO development, due to increase in the portion of transactions or components that are used repeatedly. According to [27], the impact of KBS on the main stages of CTO is very significant. Let us compare the

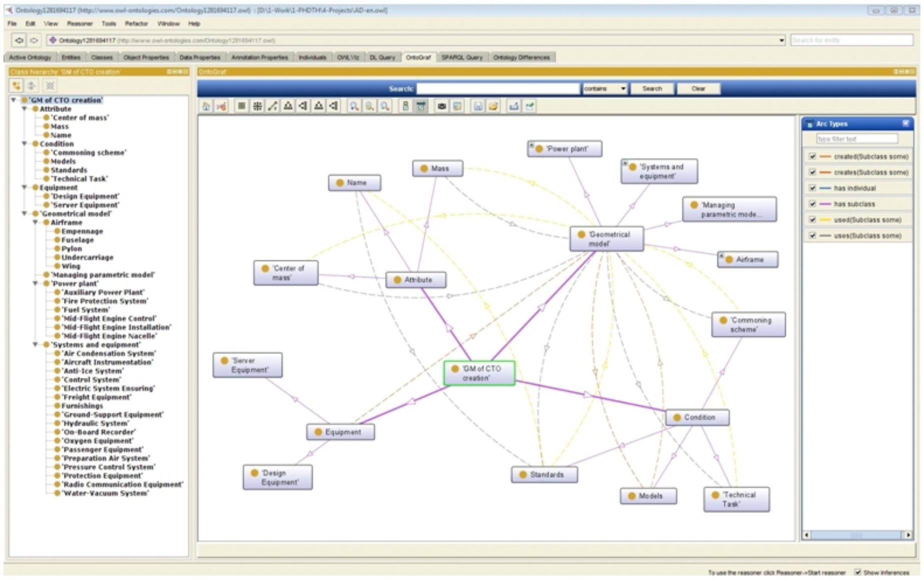


Figure 11. Ontology of CTO creation

average time of GM creation using CAx and PLM systems [1, 3] and the described KBS approach adoption of GM creation (Figure 15).

The time taken for CPM, DB and KB development, of course, is longer than the creation of MGM only, which CPM includes. Later, however, there is a significant time saving of CTO creation by reducing the duration of creating the next stages of CTO creation: SAM and CPDM. Knowledge-based modelling approach of CTO creation gives the freedom to design changes.

5 CONCLUSIONS

The tasks and features description of a complex technical object using knowledge based framework enables reliable and efficient linking of different models of CTO. The concept of CTO control parametric information was developed for the first time. Using this concept in the CTO design allows linking the various stages of design and linking together the design stage and the production of CTO by controlling the main parameters within a single CTO GM. Considering IT data exchange while creating GM in the different CAx-systems identified shortcomings of the existing methods and showed the approach of adoption of intellectualization of the exchange of data from different CAx-systems. An example of KBS adoption in CTO creation approach is presented in this paper.

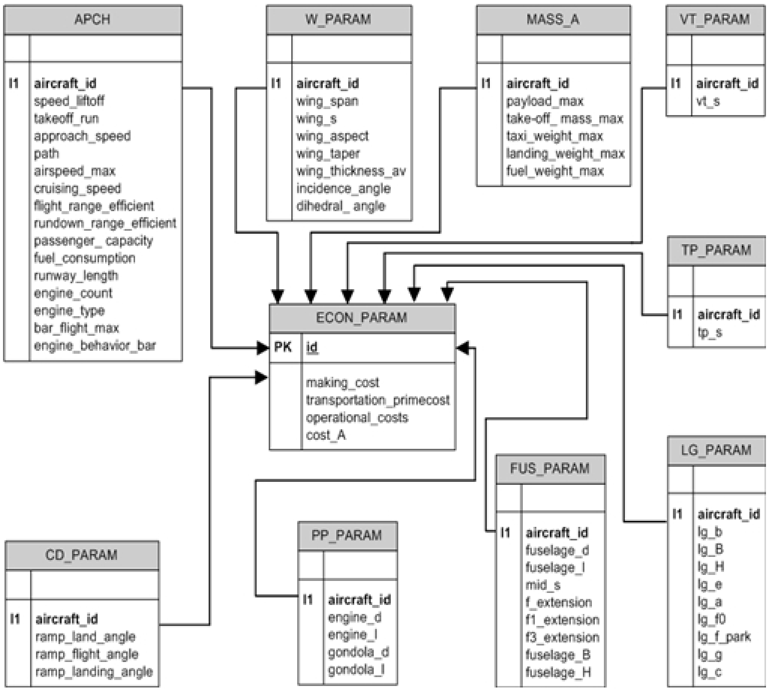


Figure 12. DB chart "CTO models parameters"

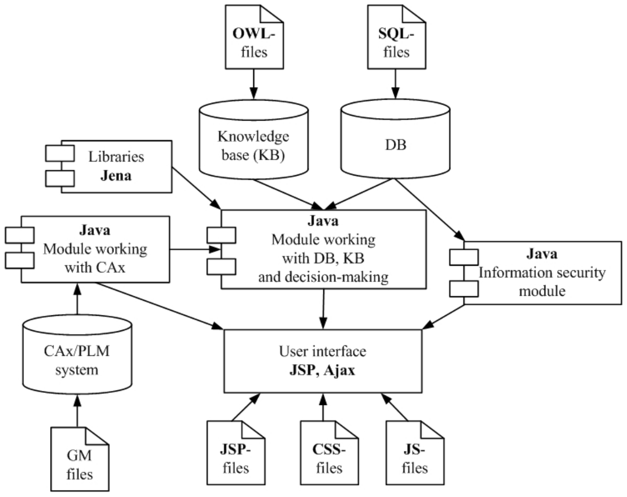


Figure 13. KBS adoption in CTO components circuit

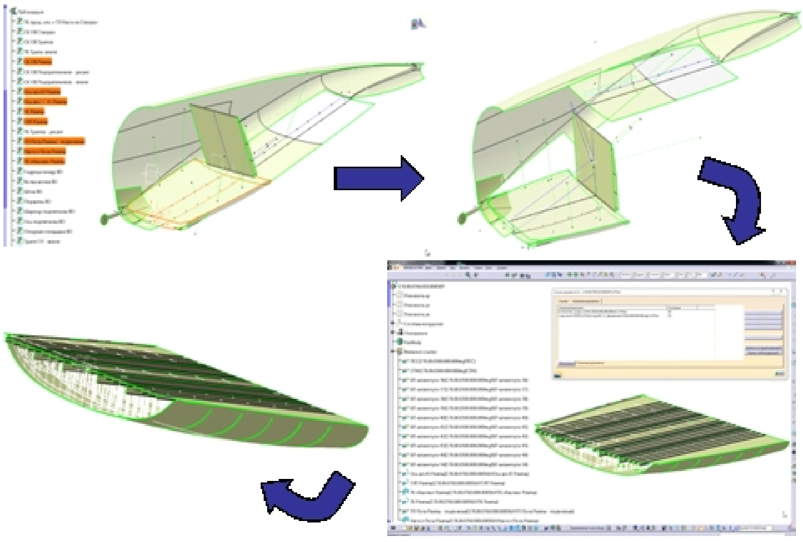


Figure 14. Data exchange from CPM to GM

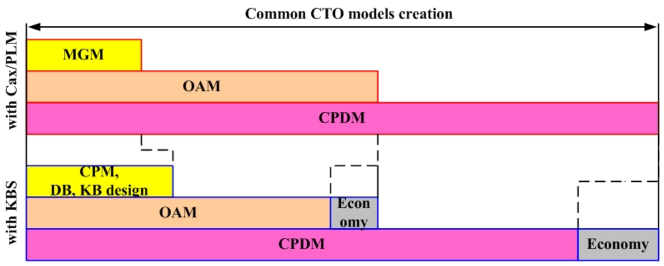


Figure 15. Comparing the average time of CTO creation using CAx, PLM systems and KBS

Acknowledgement

This work was supported by the Slovak Scientific Grant Agency VEGA under the grant No. 2/0154/16. The solemn memory of Dr. Valery Zinchenko, who managed, helped, inspired by own example to make the world better by science.

REFERENCES

[1] Raymer, D.: Aircraft Design: A Conceptual Approach. Fourth Edition, American Institute of Aeronautics and Astronautics, Reston, 2006.

[2] Konotop, D.—Budinska, I.—Zinchenko, V.—Gatial, E.: Multi-Agent-Based Conception of Modern Aircraft Design. Proceedings of the 5th Workshop on Intelligent and Knowledge Oriented Technologies, Bratislava, Slovakia, 2010, pp. 125–128.

[3] Kundu, A. K.: Aircraft Design. Queen's University Belfast, Cambridge University Press, 2010, doi: 10.1017/CBO9780511844652.

[4] Richter, T.—Mechler, H.—Schmitt, D.: Integrated Parametric Aircraft Design. ICAS 2002 Congress, International Council of the Aeronautical Sciences, 2002.

[5] Hatamura, Y. (Ed.): Decision-Making in Engineering Design: Theory and Practice. Springer, 2006, doi: 10.1007/1-84628-261-6.

[6] Stokes, M. (Ed.): Managing Engineering Knowledge: MOKA: Methodology for Knowledge Based Engineering Applications. Wiley-Blackwell, 2001.

[7] Hepp, M.—de Leenheer, P.—de Moor, A.—Sure, Y. (Eds.): Ontology Management: Semantic Web, Semantic Web Services and Business Applications. Springer, 2008.

[8] Chounta, I.-A.—Avouris, N.: Towards a Time Series Approach for the Classification and Evaluation of Collaborative Activities. Computing and Informatics, Vol. 34, 2015, No. 3, pp. 588–614.

[9] Pro-Engineer Web Site. Available at: `http://www.ptc.com/cad/pro-engineer`.

[10] CATIA Documents Website. Available at: `http://catiadoc.free.fr/online/CATIAfr_C2/kwrugCATIAfrs.htm`.

[11] PLM Automaton Siemens Web Site. Available at: `http://www.plm.automation.siemens.com/`.

[12] Hitzler, P.—Krötzsch, M.—Rudolph, S.: Foundations of Semantic Web Technologies. Chapman and Hall/CRC, 2009.

[13] Cooper, S. B.: Computability Theory. Chapman and Hall/CRC, 2003.

[14] Perakath, C. B. et al.: IDEF5 Method Report. Information Integration for Concurrent Engineering (IICE) Project, F33615-90-C-0012, Armstrong Laboratory, Ohio, 1994.

[15] Whitten, J. L.—Bentley, L. D.—Dittman, K. C.: Systems Analysis and Design Methods. McGraw-Hill Irwin, 2004.

[16] Azzolini, J.: Introduction to Systems Engineering Practices. Presentation July 2000, System Engineering Seminar. Available at: `https://ses.gsfc.nasa.gov/`, retrieved on May 2018.

[17] Laalaoui, Y.—Bouguila, N. (Eds): Artificial Intelligence Applications in Information and Communication Technologies. Springer International Publishing, Studies in Computational Intelligence, Vol. 607, 2015, doi: 10.1007/978-3-319-19833-0.

[18] Abramov, E.—Konotop, D.—Abramova, A.: Knowledge-Oriented Support of Complex Technical Object Design. Proceeding of the 2013 IEEE 2nd International Conference on Actual Problems of Unmanned Air Vehicles Developments, Kiev, Ukraine, 2013, pp. 122–125, doi: 10.1109/APUAVD.2013.6705302.

[19] IGES/PDES Organization (June 28, 2006), The Initial Graphics Exchange Specification (IGES) Version 5.x (Draft). IGES/PDES Organization. Baseline version was January 12, 1999.

[20] ISO 10303-1:1994 Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 1: Overview and Fundamental Principles.

[21] Khronos Releases Final WebGL 1.0 Specification. `https://www.khronos.org/news/press/khronos-releases-final-webgl-1.0-specification`. Retrieved 2015-05-18.

[22] Konotop, D.: Ontology Using in Geometrical Models Data Processing of Complex Technical Object. Proceedings of the XX^{th} International Conference on Knowledge-Dialogue-Solution, Kyiv, September, 8–10, 2014, pp. 118–119.

[23] Gruber, T. R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, Vol. 5, 1993, No. 2, pp. 199–220, doi: 10.1006/knac.1993.1008.

[24] Horridge, M.—Knublauch, H.—Rector, A.—Stevens, R.—Wroe, C.: A Practical Guide to Building OWL Ontologies Using the Protégé-OWL Plugin and CO-ODE Tools. Edition 1.0, The University of Manchester, 2004.

[25] Spitz, W.—Golaszewski, R.—Berardino, F.—Johnson, J.: Development Cycle Time Simulation for Civil Aircraft. NASA/CR-2001-210658, 2001.

[26] Mason, W. H.: Modern Aircraft Design Techniques. Chapter 26. Handbook of Transportation Engineering. 1st Edition. McGraw-Hill Professional, 2003.

[27] Skarka, W.: Application of MOKA Methodology in Generative Model Creation Using CATIA. Engineering Applications of Artificial Intelligence, Vol. 20, 2007, No. 5, pp. 677–690, doi: 10.1016/j.engappai.2006.11.019.

[28] Protègè Web Site. Available at: `http://protege.stanford.edu`.

[29] Paraiso, E. C.—Junior, G. B.—Ramos, M. P.—Sato, G. Y.—Tacla, C. A.: Improving Knowldge Acquisition in Collaborative Knowledge Construction Tool with Virtual Catalyst. Computing and Informatics, Vol. 35, 2016, No. 4, pp. 914–940.

[30] Qiao, L.—Qie, Y.—Zhu, Z.—Zhu, Y.—Yixin, Z.—uz Zaman, U. K.—Anwer, N.: An Ontology-Based Modelling and Reasoning Framework for Assembly Sequence Planning. The International Journal of Advanced Manufacturing Technology, Vol. 94, 2018, No. 9-12, pp. 4187–4197, doi: 10.1007/s00170-017-1077-4.

Dmytro Konotop graduated from the National Technical University of Ukraine Kyiv Polytechnic Institute (NTUU KPI) in 2008 and defended his Ph.D. thesis in the field of information technology in 2019. His research interests include information technology in the design of complex technical objects, ontology, CAD, CAM, CAE and PLM systems.

Valeriy Zinchenko graduated from the Taras Shevchenko National University of Ukraine in 1975 and he was awarded his Ph.D. in 1990. He was Assistant Professor of NTUU KPI, the winner of the State Prize of Ukraine for Science and Technology in 1994 and 2003, and he was a veteran of the Antonov aviation company. He authored more than 300 scientific papers. Valeriy Zinchenko deceased in August 2018.

Ivana Budinská graduated from the Slovak Technical University and defended her Ph.D. thesis in the field of automation at the Institute of Informatics of the Slovak Academy of Sciences. Her research interests include discrete systems modelling and simulation, multi agent systems, artificial intelligence, complex systems, and systems theory. She is an author and co-author of more than 100 research papers with above 120 citations on them.

Wei Li defended his Ph.D. thesis in the Moscow Aviation Institute (National Research University) in 2009. He is Professor and Head of Aircraft Design and Mechanics Department of Shenyang Aerospace University, People Republic of China.

Computing and Informatics, Vol. 38, 2019, 1131–1148, doi: 10.31577/cai_2019_5_1131

MOBILE EDGE COMPUTING BASED IMMERSIVE VIRTUAL REALITY STREAMING SCHEME

Juyong Lee

Department of Electronic and Information System Engineering
Sangmyung University, Cheonan, Korea
e-mail: `juyonglee0208@gmail.com`

Daeyoub Kim

Department of Information Security
Suwon University, Hwaseong-si Gyeonggi-do 18323, Korea
e-mail: `daeyoub69@suwon.ac.kr`

Jihoon Lee*

Department of Smart Information and Telecommunication Engineering
Sangmyung University, Cheonan, Korea
e-mail: `vincent@smu.ac.kr`

Abstract. Recently, new services using virtual reality (VR)/augmented reality (AR) have appeared and then exploded in entertainment fields like video games and multimedia contents. In order to efficiently provide these services to users, an infrastructure for mobile cloud computing with powerful computing capabilities is widely utilized. However, existing mobile cloud system utilizes a cloud server located at a relatively long distance, so that there are problems that a user is not effectively provided with personalized immersive multimedia service. So, this paper proposes the home VR streaming system that can provide fast content access time and high immersiveness by using mobile edge computing (MEC).

* Corresponding author

Keywords: Mobile edge computing, multi-access edge computing, mobile edge computing video service, mobile cloud computing, virtual reality streaming, video streaming

1 INTRODUCTION

Due to the widespread adoption of mobile and Internet of Things (IoT) devices, there are many emerging entertainment services where users can create and share content anytime and anywhere. Also, cloud computing such as Google Cloud Platform (GCP), Amazon Elastic Compute Cloud (EC2) and Microsoft Azure has emerged as a new computing paradigm with the explosive spread of mobile networks. Recently, thanks to the computing paradigm of cloud computing and the explosion of mobile devices, there are growing interests in new applications and services such as real-time online games, AR, VR and ultra-high definition (UHD) streaming that require very low latency and high access speeds. In the meantime, due to the need for high computing power from such immersive and high quality multimedia services, cloud computing platform has been considered as a solution to meet the service requirements.

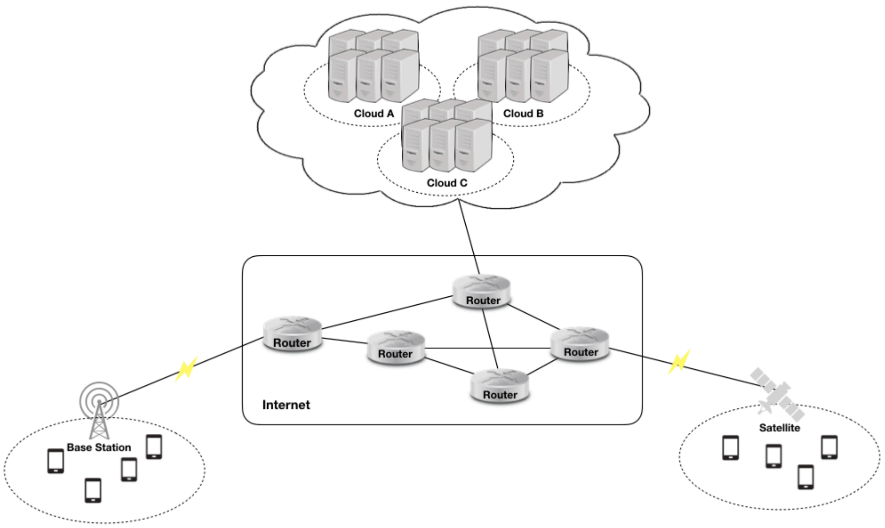


Figure 1. The existing cloud computing architecture

Even though existing cloud computing platforms have performed well computation intensive task with its powerful computing capabilities and scalability of cloud service, it has trouble supporting immersive services such as AR, VR streaming and multimedia streaming. Furthermore, the enormous volume of data exchanged between user devices and remote cloud servers cause the data tsunami, which leads to saturate and bring down the backhaul networks. In order to solve such problems, a new network architecture to place a small cloud with various computing functions close to the user devices is under active research named as Fog Computing and MEC [1, 2, 3, 4].

The concept of Fog Computing was proposed by Cisco, which is a service infrastructure that accesses the computation and storage resources [5, 6, 7]. It provides computing, storage, and a variety of application services, like the cloud, but it does not have a centralized structure on its central server. That is, Fog Computing seamlessly extends cloud computing to the edge for the secure control and management of domain specific hardware, software, storage, and network functions within the domain and enables secure rich data processing applications across the domain. Meanwhile, the concept of MEC was proposed by the European Telecommunications Standard Institute (ETSI) as a new platform that provides IT and cloud computing capabilities within a radio access network (RAN) in close proximity to mobile consumers [8, 9, 10]. So, MEC architecture can support latency-sensitive services with the backhaul capacity of limited mobile networks [11, 12, 13]. However, there is a lack of researches on the effect to mobile consumers for large capacity multimedia transmission such as VR and UHD in MEC environment.

Cloud Computing	– Central processing based model – Accessed through Internet – Easy to scale – Low cost storage
Fog Computing	– Extending cloud to the edge of the network – Decentralized computing – Realtime data analysis
Mobile Edge Computing	– Edge can work without cloud or fog – Decentralized computing – Low latency – Realtime service

Table 1. Cloud computing vs Fog computing vs Mobile edge computing

So, this paper proposes a scheme that enables users to provide low-latency VR contents streaming experience by utilizing MEC environment. The rest of the paper is organized as followed. Section 2 describes the existing cloud computing problems. Section 3 presents the proposed virtual reality streaming scheme based on MEC environment and then, we present the evaluation results in Section 4. Finally in Section 5, we conclude this paper.

2 PROBLEM STATEMENTS

In this section, we will look into the problems that arise created by the services requiring high computing power from the view point of users and cloud. First, we will look at the propagation distance of cloud computing caused by the centralized structure and also the effects of when multiple users try to access the cloud server. In addition, we will describe the resource management problem in the cloud environment. Finally, we show the energy efficiency problem created by the services requiring high computing resources.

2.1 Propagation Distance and Multi User Access Problem

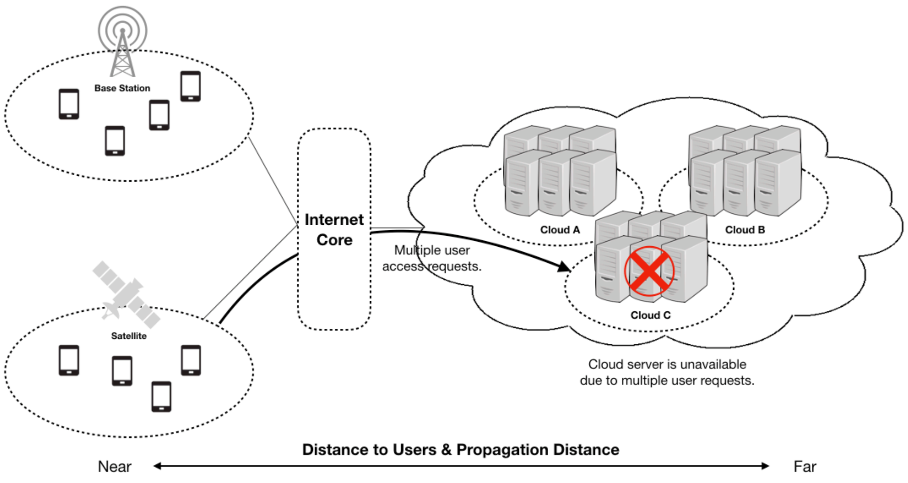

Base Station
Satellite
Internet Core
Multiple user access requests.
Cloud A
Cloud B
Cloud C
Cloud server is unavailable due to multiple user requests.
Distance to Users & Propagation Distance
Near
Far

Figure 2. Propagation distance and multi user access problem

For immersive multimedia content, the propagation distance, which affects the transmission latency, is very critical. However, in existing mobile cloud computing environment, mobile consumer devices need to deliver or download their desired data from remotely located servers in the data center or core network. The centralized mobile cloud structure creates many challenges. First, it brings about long latency because various types of data are transmitted across multiple networks including wireless access networks, backhaul networks, and the Internet, which require its own traffic control, routing, and various network management tasks. Second, as the number of mobile consumer devices connected to the cloud server increases, the network resources such as the bandwidth are excessively used, making it impossible to deliver contents to the content requester quickly. Finally, the mobile cloud computing needs to share its computing resources with a much larger number of mobile consumer devices, which aggravates the computation latency due to increased processing delay and high sharing of system resource. Therefore, it leads to the situation that it cannot be assured of a low-latency content transmission as the content requestor is located at a distance from content source.

2.2 Cloud Computing Resource and Task Management Problem

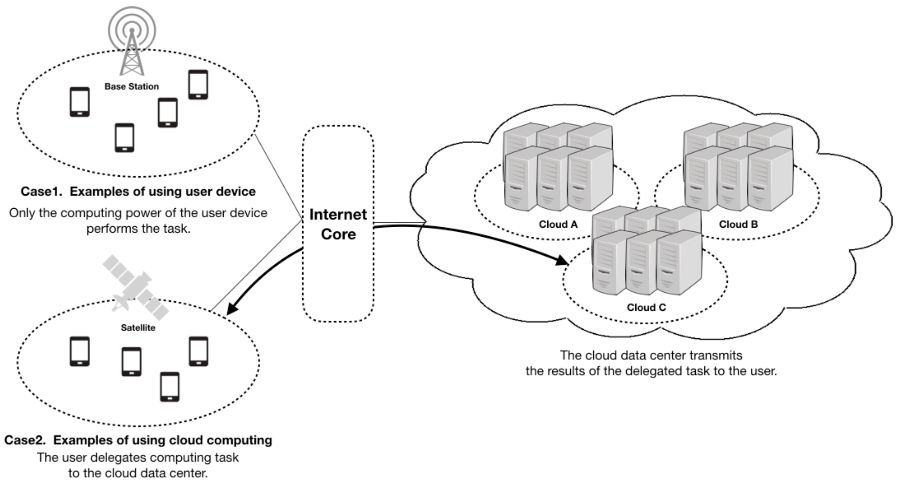

Base Station
Case1. Examples of using user device
Only the computing power of the user device performs the task.
Internet Core
Satellite
Case2. Examples of using cloud computing
The user delegates computing task to the cloud data center.
Cloud A
Cloud B
Cloud C
The cloud data center transmits the results of the delegated task to the user.

Figure 3. Cloud computing resource and task management problem

Immersive multimedia services require not only many computationally intensive tasks (i.e., encoding, decoding, and transcoding, etc.) but a large storage capacity. Moreover, computing-related services that require robust memory and powerful computing performance (i.e., machine learning, image object detection, video transcoding, live streaming, etc.) lead to high energy consumption. However, sufficient energy supply is required to meet the above-mentioned requirements in order to use these services, but there is a great limitation in mobile and IoT devices using resource constrained batteries. Because of their compact size, mobile and IoT devices have limited energy capacity; therefore, computing power and high energy consumption are key factors. To solve these problems, cloud computing systems use a method of delegating tasks to be performed on mobile devices to a cloud having powerful computing functions. That is, existing cloud computing services are operated in a way that users delegate their own processing to a data center that is rich in computing resources specific to a task.

However, due to the characteristics of this cloud computing architecture, two problems arise; computing resource management and task migration problem. There is a possibility for unnecessary energy consumption, resulting from wasted computing resources, since the provision of services suited to the characteristics of the cloud server and the efficient server arrangement structure are not considered [14, 15, 16]. So, excessive energy consumption and inefficient processing overhead happen when a large number of user's service requests are received because existing schemes adopt a fixed resource allocation policy for user requests. Therefore, the computing resources are not properly allocated and long service latency is caused. This has become a critical factor for multimedia services.

2.3 Mobile Energy Efficiency Problem

Immersive multimedia services such as VR and AR require high computing power as well as a lot of computation-intensive tasks (i.e., encoding, decoding, transcoding, and head tracking, etc.), which results in high energy consumption. For mobile and IoT devices, high energy consumption is a critical factor. That is, running high computationally demanding applications at mobile user devices is constrained by limited battery capacity and energy consumption of the mobile user devices. In order to solve such a problem, a method of collecting and transmitting data without repeated transmission was proposed [17, 18, 19, 20].

However, these methods are not appropriate for both IoT devices and real-time services in which up-to-date information needs to be managed and monitored. In addition, although this solution can reduce the energy required for network transmission, efficient energy usage cannot be achieved while intensive computing services are used due to limited constrained energy when performing computationally intensive computing operations.

3 VIRTUAL REALITY STREAMING SCHEME BASED ON MOBILE EDGE COMPUTING ENVIRONMENT

To solve the structural limitations of existing cloud computing systems and at the same time, enhance the utilization efficiency of computing resources (i.e., memory, storage, network bandwidth, and so on), the proposed scheme has the hybrid scheme that utilizes both cloud computing and MEC simultaneously. As shown in Figure 4, the proposed scheme assumes the MEC environment to provide low latency VR content streaming to mobile consumer devices. The localized mobile edge computing server (MECS) is a key component to VR content encoding and transmission in the proposed scheme. That is, the MECS located at the network edge takes over performing computation-intensive tasks to be originally handled by the mobile consumer and IoT devices, thereby reducing the computing operation and service provision latency as well as allowing the service provision near the mobile consumer device. The proposed scheme starts by periodically transmitting the network status of wireless access point (WAP) to the relevant MECS in order to transmit the VR content quickly. That is, MECS recognizes the network status and then helps to select the MECS that best suits the specific task. The proposed scheme consists of two methods: live streaming of VR content and streaming of stored VR content.

3.1 Live Streaming of VR Content

The proposed scheme starts when the content source, that wants to stream VR contents in real time, sends a content streaming request message to its access MECS.

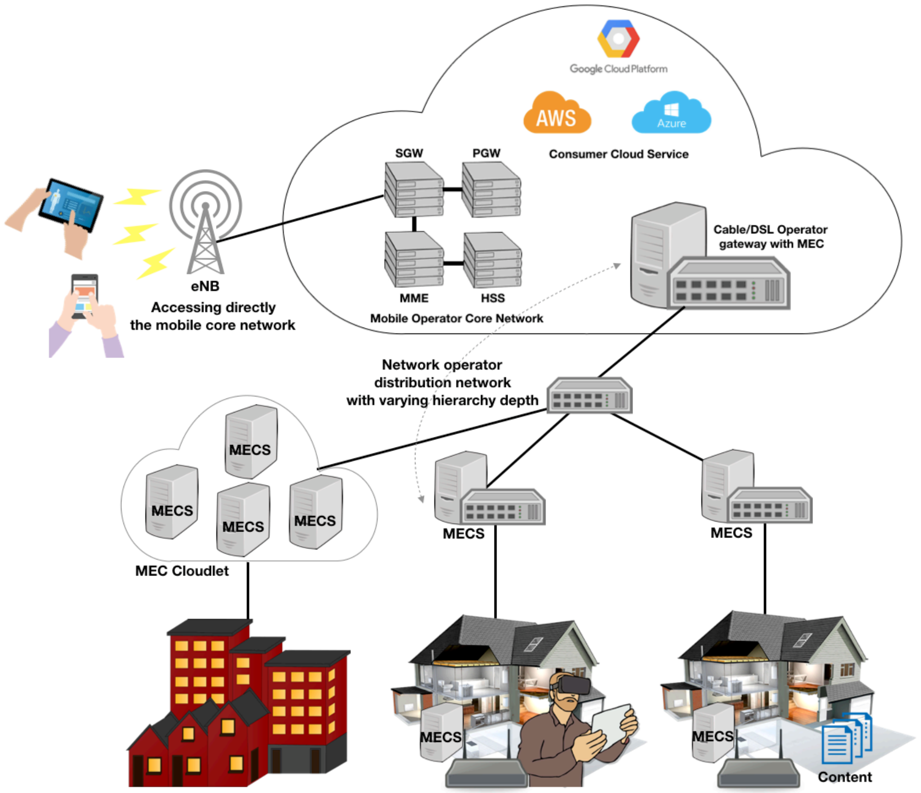

Google Cloud Platform
AWS
Azure
Consumer Cloud Service
SGW
PGW
MME
HSS
Mobile Operator Core Network
Cable/DSL Operator gateway with MEC
eNB
Accessing directly the mobile core network
Network operator distribution network with varying hierarchy depth
MECS
MECS
MECS
MECS
MEC Cloudlet
MECS
MECS
MECS
MECS
Content

Figure 4. Proposed service architecture for virtual reality streaming scheme

The access MECS receiving the message from the content source performs the task of selecting the MECS suitable for VR streaming by checking the computing resources and network status of the neighboring MECs. The proposed approach delegates computation-intensive task to the neighboring MEC when excessive computation-intensive task is concentrated on MECS. Since network and computing capacity status of each MECS are monitored in advance by exchanging information beforehand among MECSs, each MECS can make a decision on encoding process. In addition, when the state of each MECS's network and computing resource is changed due to the user's VR streaming request, the MECS transmits its resource variation information to the Broker.

Upon receiving the request, the access MECS determines whether VR content encoding and streaming is possible using its own computing resources. If the access MECS can encode and stream VR content, it provides encoding and streaming service to users using its own computing resources and transmits to the Broker its streaming information to generate a content list. In contrast, if the access MECS

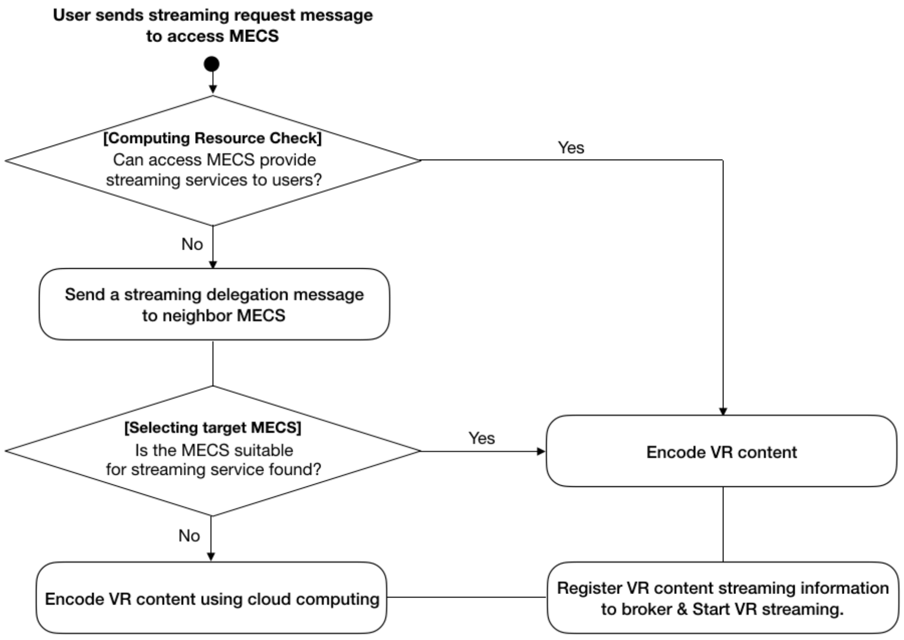

User sends streaming request message to access MECS
[Computing Resource Check] Can access MECS provide streaming services to users?
Yes
No
Send a streaming delegation message to neighbor MECS
[Selecting target MECS] Is the MECS suitable for streaming service found?
Yes
No
Encode VR content
Encode VR content using cloud computing
Register VR content streaming information to broker & Start VR streaming.

Figure 5. The operation procedures of the proposed MECS based VR live streaming service

cannot provide the service, it selects the serviceable MECSs using the status information of the neighboring MECs and the Broker, and transmits the VR streaming delegation message to the selected MECs. The neighboring MECS receiving the VR content streaming delegation request message from the access MECS performs VR content encoding and streaming service using its own computing resources. Figure 5 shows the operation procedures of the proposed MECS based on VR live streaming service. That is, since the proposed scheme utilizes the MEC architecture based on the computing power, it can efficiently cope with a large number of VR streaming requests through the resources of the MECS. If the neighboring MECS suitable for the VR streaming operation is selected, the access MECS delegates the VR content encoding and streaming operation to the selected MECS. Therefore, as the proposed method delegates the task processing to a specific MECS that has enough computing capacity and network resources, it enables providing low service latency and improved energy efficiency to mobile consumer device. The rest of the process is the same as using existing cloud-based normal streaming services. That is, since the MECS provides the encoding and streaming service, the operation flow is simplified and the processing burden of the mobile device is reduced.

Figure 6 shows how the proposed scheme works for VR encoding and streaming services. First, mobile user sends VR content streaming request message to the

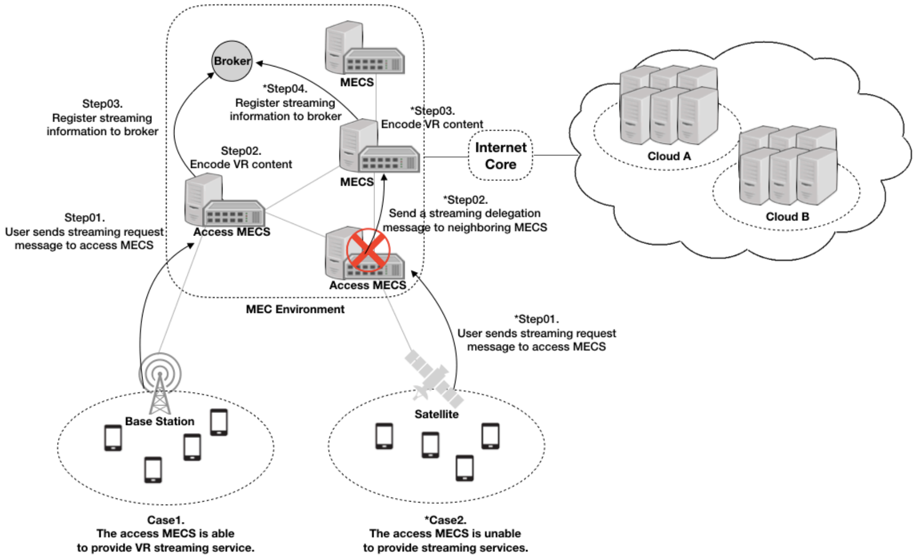

Broker
MECS
*Step04. Register streaming information to broker
*Step03. Encode VR content
Step03. Register streaming information to broker
Step02. Encode VR content
Internet Core
Cloud A
Cloud B
MECS
*Step02. Send a streaming delegation message to neighboring MECS
Step01. User sends streaming request message to access MECS
Access MECS
Access MECS
MEC Environment
*Step01. User sends streaming request message to access MECS
Base Station
Satellite
Case1. The access MECS is able to provide VR streaming service.
*Case2. The access MECS is unable to provide streaming services.

Figure 6. An example of live streaming of VR content in the proposed scheme

access MECS. After receiving the VR content streaming request message, the MECS decides whether it can process the VR streaming service. If the MECS can handle the service request, it is in charge of the service request. Otherwise, it forwards the VR streaming request to neighboring MECS. After that, the mobile user wanting to deliver live VR streaming content simply streams the VR content to the MECS without encoding it on his/her own mobile device. Before watching the VR content, each content consumer accesses the VR content list after a registration/login process and then selects VR content to view. After that, the MECS delivers encoded VR content to the content consumer device. As the MECS is responsible for encoding and streaming VR content, the overall operation is simplified and at the same time, the processing burden of the mobile consumer device is reduced. Moreover, as the MECS is located in a relatively short distance from VR streaming source, it leads to low service latency critical to immersive VR content delivery.

3.2 Streaming of Stored VR Content Files

To store VR content in the cloud data center and to share them with other consumers, the proposed scheme is composed of two kinds of use cases as shown in Figure 7. The proposed scheme for streaming stored VR content files uses existing cloud computing architecture and MEC environment. Part 1 covers with the encoding of VR content in a distributed MECS environment.

In part 1, the mobile user trying to record specific VR content first transmits the VR content to its access MECS. The access MECS receiving the VR content

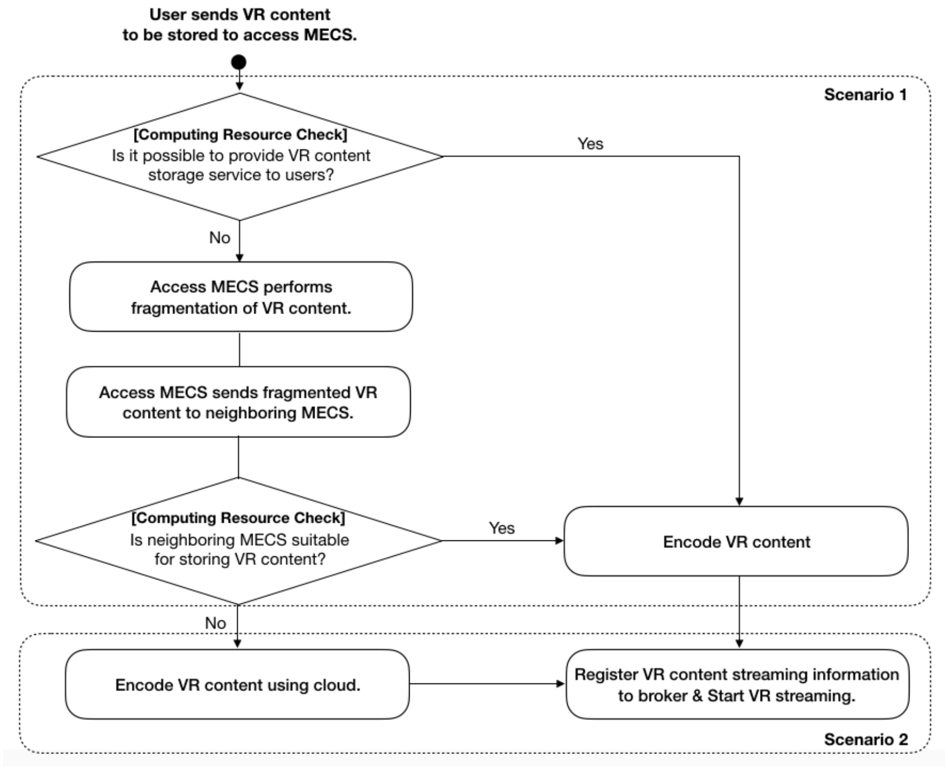

User sends VR content
to be stored to access MECS.
Scenario 1
[Computing Resource Check]
Is it possible to provide VR content
storage service to users?
Yes
No
Access MECS performs
fragmentation of VR content.
Access MECS sends fragmented VR
content to neighboring MECS.
[Computing Resource Check]
Is neighboring MECS suitable
for storing VR content?
Yes
Encode VR content
No
Encode VR content using cloud.
Register VR content streaming information
to broker & Start VR streaming.
Scenario 2

Figure 7. The operation procedures to stream stored VR content in the proposed scheme

performs a task to check whether that its own computing resources are sufficient to perform the VR content fragmentation operation. If the computing resources of the access MECS are sufficient, it performs the VR content encoding task itself. If the size of the VR content file is large, the VR content is fragmented and the encoding task for the VR content is delegated to the neighboring MECS. After VR content fragmentation, the neighboring MECS, to which the encoding task is delegated, encodes the fragmented VR content and then forwards the encoded VR content to the central cloud server. Since the proposed scheme can work encoding VR content together with neighboring MECS, computing resources are efficiently utilized and at the same time, the encoding is completed fast. Also, the central cloud server simply acts as a distribution point for the cached VR content. All fragmented VR content is stored at the data center in the central cloud and then is consumed at the subsequent requests. Moreover, the proposed scheme assumes the popularity-driven content caching policy for low latency service provision. That is, popular VR content is cached at local MECS. After that, when the stored VR content is requested, it is delivered from local MECS, not from the central cloud, which results in low service latency and high immersiveness. In other words, the proposed scheme can provide multiple content deliveries from both the remotely-located cloud data center and the local MECS near the content consumer so that low service latency and content quality are guaranteed when compared with the existing cloud computing architecture.

On the other hand, part 2 indicates when the requested VR content is received from the repository of the central cloud due to the lack of computing resources at local MECS. In other words, if access MECS and neighboring MECS cannot meet the service requests from multiple content consumers, the requested content is forwarded from the cloud data center. The main difference between the proposed scheme and existing cloud scheme is to provide low service latency by localized transmission through MECS's caching functionality. Whenever content is delivered toward content consumers, it is cached at MES, which leads to lower network resource utilization and lower service latency than conventional cloud computing platforms.

Figure 8 shows an example of saving and streaming VR content. User sends VR content to his/her access MECS to store VR content. The access MECS receiving the VR content performs fragmentation of the VR content. After completing the VR content fragmentation, the access MECS sends the fragmented VR content to the neighboring MECS. When receiving the fragmented VR content, the neighboring MECS performs the encoding operation and transmits the encoded file to the cloud data center and the neighboring MECS. By caching the encoded VR content at the local MECS, the proposed scheme can distribute the VR content faster than the existing cloud computing architecture.

4 PERFORMANCE EVALUATION

To evaluate the effectiveness of the proposed scheme, the computing power consumption, average service latency and the content download time are measured against various resolutions of VR content and content size, respectively.

4.1 VR Content Download Time

In order to compare the content download time of the proposed scheme with existing cloud architecture, variable sizes of VR content are considered. The MECS function is implemented by using Docker virtualization image [21]. Figure 9 shows the VR content download time against various sizes of VR content. When VR content size is small, the service delivery time is not significantly different between central cloud data center with far distance and the proposed scheme because the transmission distance is not a critical factor due to low volume of the content data. However, as shown in Figure 9, as the number of mobile users at the edge networks and the number of content requests for VR content increases, congestion of the core network and edge network occurs, and therefore an additional operation such as content migration operation must be performed, it may take a long download time.

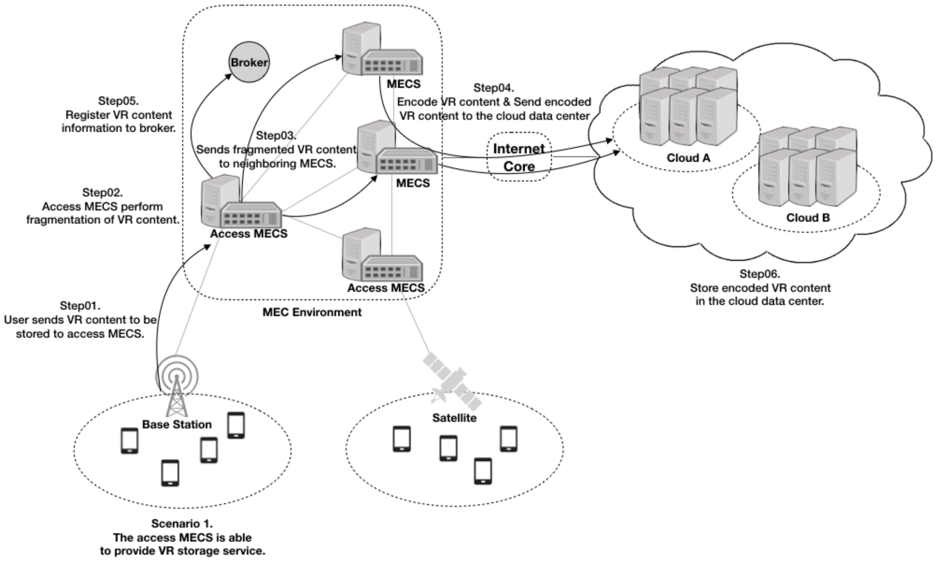

Broker
MECS
Step04.
Encode VR content & Send encoded VR content to the cloud data center
Step05.
Register VR content information to broker.
Step03.
Sends fragmented VR content to neighboring MECS.
Internet Core
Cloud A
MECS
Cloud B
Step02.
Access MECS perform fragmentation of VR content.
Access MECS
Access MECS
Step06.
Store encoded VR content in the cloud data center.
Step01.
User sends VR content to be stored to access MECS.
MEC Environment
Satellite
Base Station
Scenario 1.
The access MECS is able to provide VR storage service.

Figure 8. Operation procedure for storing VR content and delivering stored VR content in the proposed scheme

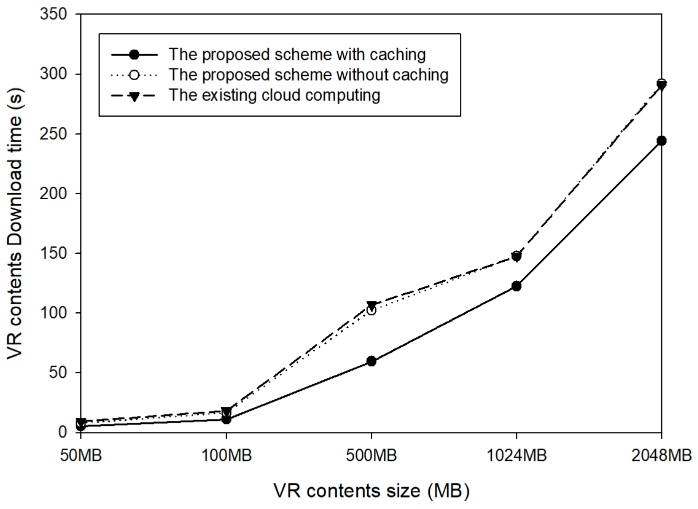


Figure 9. VR content download time

Also, since the proposed scheme uses the local MEC caching function and the cloud computing architecture, it is possible to provide VR contents faster than the existing cloud and MEC environment. Meanwhile, as the VR content size increases, the existing cloud computing architecture experiences longer transmission delay due to long distance between content consumer and the central cloud server. In addition, the proposed scheme utilizes cached content in a localized MECS, which leads to shorter transmission delay. In other words, the proposed scheme provides the local accesses to the VR content and delivers the cached data from the access MECS locally located at the content consumer.

4.2 Average Response Latency

Figure 10 shows the average response latency when a user requests VR streaming services to a server based on the number of users. The response latency is assumed to exclude VR encoding time done after receiving VR streaming service request. As the number of users who request VR streaming services to the server increases, existing cloud computing architecture has significantly higher latency than the proposed scheme. It comes down to the fact that the data centers in existing cloud architecture are remotely located from the users. Meanwhile, the proposed scheme can provide low response latency because much processing load are performed at MECS close to users. Moreover, as the proposed scheme tries to delegate VR streaming tasks to neighboring MECSs, it can keep a low response latency no matter what the number of user request increases.

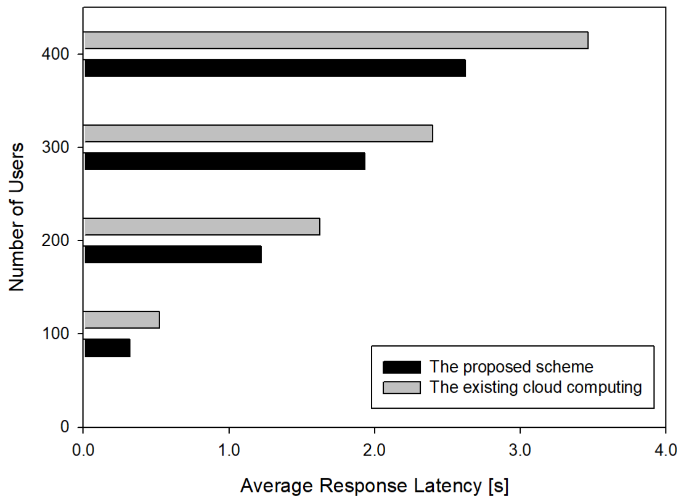


Figure 10. Average response latency

4.3 Energy Consumption

Finally, let us focus on the system efficiency of the proposed scheme in terms of the energy consumption. The energy consumption is measured in Watts unit according to the encoding types of 240p, 480p, and 1080p VR content, respectively. Figure 11 shows the computing power consumed to encode each VR content. When VR content resolution is low, the computation intensive task of the VR content encoding is not so large. So, a relatively small amount of computation is required, which consumes less computing power. However, as VR content resolution is high, the computation intensive task increases by VR content encoding, thus leading to much more power consumption. Meanwhile, the proposed scheme performs VR encoding by distributing Docker images based on virtualization using MEC environment. In other words, VR encoding is performed by allocating the appropriate resources required for VR encoding, thus enabling efficient utilization of computing resources. Therefore, the proposed scheme can reduce the workload such as setting up computing resources and task delegation. From that, it can reduce the energy consumption rate when compared to the existing cloud architecture.

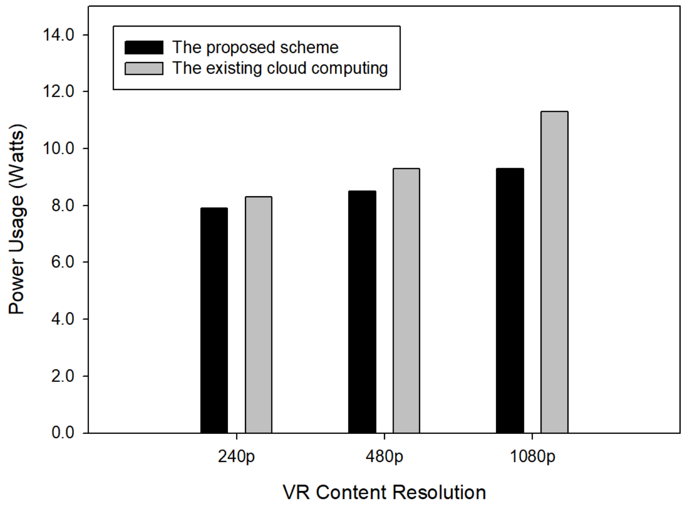


Figure 11. Computing power consumption against various VR content size

5 CONCLUSION

This paper shows the following main points. First, the existing central cloud computing architecture cannot fully support real-time applications such as VR streaming/gaming because the central cloud servers art mostly far apart from content consumers. Moreover, high volume VR streaming content requires a lot of computation load for content encoding. It is critical to mobile consumer devices with battery lifetime constraint. Second, to reduce energy consumption and service latency of consumer devices, the proposed scheme uses MEC that can provide powerful computing power in a close location to the user. It provides low latency service by virtue of its proximate position to the mobile user, and also reduces the computing load of content consumers by delegating to MECS the high intensive computation tasks like VR content encoding. Moreover, it extends lifetime of resource-constrained mobile/IoT devices. Therefore, the proposed scheme can support immersive services requiring fast access time and ultra-high computing performance.

6 ABBREVIATIONS

The following abbreviations are used in this manuscript.

VR: Virtual Reality

AR: Augmented Reality

GCP: Google Cloud Platform

EC2: Amazon Elastic Compute Cloud

IoT: Internet of Things

UHD: Ultra-High Definition

ETSI: The European Telecommunications Standard Institute

RAN: Radio Access Network

MEC: Mobile Edge Computing

MECS: Mobile Edge Computing Server

WAP: Wireless Access Point

Acknowledgement

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (grants No. 2016R1D1A1B03931628 and No. 2017R1D1A1B03034215).

REFERENCES

[1] Bonomi, F.—Milito, R.—Zhu, J.—Addepalli, S.: Fog Computing and Its Role in the Internet of Things. Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing (MCC '12), ACM, 2012, pp. 13–16, doi: 10.1145/2342509.2342513.

[2] Al-Fuqaha, A.—Guizani, M.—Mohammadi, M.—Aledhari, M.—Ayyash, M.: Internet of Things: A Survey on Enabling Technologies, Protocols,

and Applications. IEEE Communications Surveys and Tutorials, Vol. 17, 2015, No. 4, pp. 2347–2376, doi: 10.1109/COMST.2015.2444095.

[3] Stojmenovic, I.—Wen, S.: The Fog Computing Paradigm: Scenarios and Security Issues. In: Ganzha, M., Maciaszek, L., Paprzycki, M. (Eds.): Proceedings of the 2014 Federated Conference on Computer Science and Information Systems (FedCSIS). IEEE, Annals of Computer Science and Information Systems, Vol. 2, 2014, pp. 1–8, doi: 10.15439/2014F503.

[4] Lee, J. Y.—Kim, D. Y.—Lee, J. H.: ZONE-Based Multi-Access Edge Computing Scheme for User Device Mobility Management. Applied Sciences, Vol. 9, 2019, No. 11, pp. 2308–2323, doi: 10.3390/app9112308.

[5] Bonomi, F.—Natarajan, P.—Milito, R.—Zhu, J.: Fog Computing: A Platform for Internet of Things and Analytics. In: Bessis, N., Dobre, C. (Eds.): Big Data and Internet of Things: A Roadmap for Smart Environments. Springer, Cham, Studies in Computational Intelligence, Vol. 546, 2014, pp. 169–186, doi: 10.1007/978-3-319-05029-4_7.

[6] Vaquero, L. M.—Rodero-Merino, L.: Finding Your Way in the Fog: Towards a Comprehensive Definition of Fog Computing. ACM SIGCOMM Computer Communication Review, Vol. 44, 2014, No. 5, pp. 27–32, doi: 10.1145/2677046.2677052.

[7] Jennings, B.—Stadler, R.: Resource Management in Clouds: Survey and Research Challenges. Journal of Network and Systems Management, Vol. 23, 2015, No. 3, pp. 567–619, doi: 10.1007/s10922-014-9307-7.

[8] Botta, A.—de Donato, W.—Persico, V.—Pescapé, A.: On the Integration of Cloud Computing and Internet of Things. 2014 International Conference on Future Internet of Things and Cloud (FiCloud), IEEE, 2014, pp. 23–30, doi: 10.1109/FiCloud.2014.14.

[9] Lee, J. Y.—Lee, J. H.: Mobile Edge Computing Based Charging Infrastructure Considering Electric Vehicle Charging Efficiency. Journal of the Korea Academia-Industrial Cooperation Society, Vol. 18, 2017, No. 10, pp. 669–674, doi: 10.5762/KAIS.2017.18.10.669.

[10] Hu, Y. C.—Patel, M.—Sabella, D.—Sprecher, N.—Young, V.: Mobile Edge Computing – A Key Technology Towards 5G. ETSI White Paper No. 11, 2015, pp. 1–16.

[11] Roman, R.—Lopez, J.—Mambo, M.: Mobile Edge Computing, Fog et al.: A Survey and Analysis of Security Threats and Challenges. Future Generation Computer Systems, Vol. 78, 2018, Part 2, pp. 680–698, doi: 10.1016/j.future.2016.11.009.

[12] Mao, Y.—You, C.—Zhang, J.—Huang, K.—Letaief, K. B.: A Survey on Mobile Edge Computing: The Communication Perspective. IEEE Communications Surveys and Tutorials, Vol. 19, 2017, No. 4, pp. 2322–2358, doi: 10.1109/COMST.2017.2745201.

[13] Lee, J. Y.—Lee, J. H.: Hierarchical Mobile Edge Computing Architecture Based on Context Awareness. Applied Sciences, Vol. 8, 2018, No. 7, pp. 1160–1171, doi: 10.3390/app8071160.

[14] Yu, W.—Liang, F.—He, X.—Hatcher, W. G.—Lu, C.—Lin, J.—Yang, X.: A Survey on the Edge Computing for the Internet of Things. IEEE Access, Vol. 6, 2018, pp. 6900–6919, doi: 10.1109/ACCESS.2017.2778504.

[15] Tran, T. X.—Hajisami, A.—Pandey, P.—Pompili, D.: Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges. IEEE Communications Magazine, Vol. 55, 2017, No. 4, pp. 54–61, doi: 10.1109/MCOM.2017.1600863.

[16] Wang, S.—Zhang, X.—Zhang, Y.—Wang, L.—Yang, J.—Wang, W.: A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications. IEEE Access, Vol. 5, 2017, pp. 6757–6779, doi: 10.1109/ACCESS.2017.2685434.

[17] Lee, J. Y.—Lee, J. H.: Pre-Allocated Duplicate Name Prefix Detection Mechanism Using Naming Pool in CCN Based Mobile IoT Networks. Mobile Information Systems, Vol. 2016, 2016, Art. No. 9684032, 9 pp., doi: 10.1155/2016/9684032.

[18] Truong, H.-L.—Dustdar, S.: Principles for Engineering IoT Cloud Systems. IEEE Cloud Computing, Vol. 2, 2015, No. 2, pp. 68–76, doi: 10.1109/MCC.2015.23.

[19] Wang, T.—Zhang, G.—Liu, A.—Bhuiyan, M. Z. A.—Jin, Q.: A Secure IoT Service Architecture with an Efficient Balance Dynamics Based on Cloud and Edge Computing. IEEE Internet of Things Journal, Vol. 6, 2019, No. 3, pp. 4831–4843, doi: 10.1109/JIOT.2018.2870288.

[20] Muhammad, G.—Rahman, S. M. M.—Alelaiwi, A.—Alamri, A.: Smart Health Solution Integrating IoT and Cloud: A Case Study of Voice Pathology Monitoring. IEEE Communications Magazine, Vol. 55, 2017, No. 1, pp. 69–73, doi: 10.1109/MCOM.2017.1600425CM.

[21] Boettiger, C.: An Introduction to Docker for Reproducible Research. ACM SIGOPS Operating Systems Review, Vol. 49, 2015, No. 1, pp. 71–79, doi: 10.1145/2723872.2723882.

Juyong Lee received his B.Sc. and M.Sc. degrees in information and telecommunication engineering from Sangmyung University, Korea in 2014 and 2016, respectively. He is currently a Ph.D. candidate in electronic information system engineering at Sangmyung University, Korea. His research interests include multi-access edge computing, content centric networking, mobile cloud computing, future internet, software defined networking, and network security.

Daeyoub Kim received his B.Sc., M.Sc., and Ph.D. degrees in mathematics from Korea University, Seoul, in 1994, 1997, and 2000, respectively. From 1997 to 2002, he worked for Telemann and Secui.Com as a researcher. From 2002 to 2011, he was a senior research member in Samsung Electronics. In 2012, he joined the University of Suwon, Korea, where he is currently Associate Professor in the Department of Information Security. His research interests include DRM/CAS, home networks, and Internet security.

Jihoon Lee received his B.Sc., M.Sc., and Ph.D. degrees in electronics engineering from Korea University, Seoul, Korea in 1996, 1998, and 2001, respectively. From 2002 to 2011 he worked at Samsung Electronics as a senior research member. He is currently Associate Professor at the Department of Smart Information and Telecommunication Engineering, Sangmyung University. His research interests include information centric networking, context-aware networking, mobile cloud, and network security.

Computing and Informatics, Vol. 38, 2019, 1149–1180, doi: 10.31577/cai_2019_5_1149

FORMAL VERIFICATION OF SECURITY PATTERN COMPOSITION: APPLICATION TO SCADA

Fadi Obeid, Philippe Dhaussy

Lab-STICC CNRS, UMR 6285
Ensta Bretagne, 2 Rue François Verny
29200 Brest, France
e-mail: {fadi.obeid, philippe.dhaussy}@ensta-bretagne.fr

Abstract. Information security was initially required in specific applications, however, nowadays, most companies and even individuals are interested in securing their information assets. The new requirement can be costly, especially with the high demand on security solutions and security experts. Security patterns are reusable security solutions that prove to be efficient and can help developers achieve some security goals without the need for expertise in the security domain. Some security pattern combinations can be beneficial while others are inconsistent. Model checking can be used to verify the production of combining multiple security patterns with an architecture. Supervisory control and data acquisition (SCADA) systems control many of our critical industrial infrastructures. Due to their limitations, and their augmented connectivity, SCADA systems have many unresolved security issues. In this paper, we demonstrate how we can automatically generate a secure SCADA model based on an insecure one and how to verify the generated model.

Keywords: Information security, security patterns, formal verification, model checking, SCADA

1 INTRODUCTION

Information security is a tremendous challenge, whereas meeting security requirements without affecting the services is a difficult task. General security guidelines [20] are useful in avoiding many security problems and building a fairly secure system. Furthermore, some security problems are highly recurrent and efforts to resolve them are redundant. A best solution may be found in a specific context which can later be beneficial for others. A security pattern describes how to apply a specific security measure to solve a security problem.

To achieve multiple security goals, multiple security patterns can be used. However, while some security patterns are better when combined, others are inconsistent and can raise new problems. Combining multiple security patterns in an application can have additional issues depending on the application itself.

An efficient security solution should solve a security issue to fulfill some security requirements without affecting other system requirements. Model checking can be used to verify that the system meets its requirements and ensures correct functionality [6]. It can also verify the consistency of the security patterns with the architecture and with each other. This is important to validate the security properties of the system as well as the security properties of the patterns themselves.

Supervisory control and data acquisition (SCADA) systems control many of our critical industrial infrastructures, such as nuclear and chemical plants. Due to their limitations, and their augmented connectivity, SCADA systems raise tremendous challenges for security experts. With their requirements and the criticality of their security, it is very difficult to achieve a good security level in SCADA systems. Finally, attacks on SCADA have risen lately [5].

The main objective of our work is to automatically generate a secure architecture based on an insecure architecture and some security requirements. We also need to verify the result using model checking to make sure the security requirements of the architecture are respected. This requires three different elements:

1. a library of security patterns,
2. a tool to automatically combine the patterns with an architecture,
3. a model-checker to verify the resulting model.

The effiency of the patterns and the choice of the security policies are out of the scope of this paper.

In this paper we demonstrate how we can automatically generate a secure SCADA model based on an insecure one. The generated model applies multiple security patterns based on a security policy. The model can later be verified using model checking to validate the security properties and ensure correct functionality. Finally, we present some measurements regarding the complexity of our approach.

We start by introducing a background study of related work in Section 2. We specify the abstract model in Section 3, any implementation of our approach should respect this model. In Section 4, we illustrate the included security patterns and their security properties. We define our approach for security patterns combination in Section 5. The verification approach is explained in Section 6. In Section 7, we describe our implementation approach along with an example and some complexity studies. Finally, we conclude in Section 8.

2 RELATED WORK

Yoshioka et al. [26] offer a survey on security patterns. Many security patterns were introduced by Yoder and Barcalow [25] and Fernandez and Pan [14]. Schumacher et al. [21] provide a full description of integrating security patterns into systems. Hafiz et al. [17] present a pattern language unifying and classifying all published security patterns at the time. Wassermann and Cheng [24] detail many security patterns and enable the verification of these patterns by adding formal constraints to them.

Avalle et al. [3] wrote a survey on formal verification of security protocol implementations, focusing on the automatic verification of models close to the real implementation. There are many research efforts on model checking of design patterns [2, 10, 1, 22] which we consider enriching. Dong et al. [11] investigate model checking of security pattern combinations where the authors explain how wrongly combining security patterns may result in several errors.

Concerning SCADA security, many studies [19, 18, 27] were conducted to address the security problems in SCADA and/or give some theoretical solutions and guides for improving SCADA security. Other studies [23, 15] aim to enhance the security level and fortify the provided services along with the managed critical data. SCADA-specific security solutions and SCADA-specific IDS are proposed by Fovino et al. [16] and Zhu and Sastry [28], respectively. Fernandez et al. [12] propose the use of security patterns to design secure SCADA systems.

To the best of our knowledge, there is no research work that considers formally verifying properties on an auto generated architecture with a combination of security patterns. In our work, we are interested in generating secure architecture (using security patterns) and providing automatic proofs of the robustness of the generated architecture based on specific security requirements.

3 ABSTRACT MODEL

Figure 1 describes the abstract model on which security patterns are applied. We consider an architecture containing multiple entities (*Entity*) communicating between each other through communication channels (*Channel*). The communication channel has a *Fifo* to organize the order of messages (*Message*). The entities are divided into two categories: *Components* which are the assets of the architecture and need to be secured, and *Clients* that are seen as guests to the architecture. Any entity can receive and send messages. The client (*Client*) can additionally initiate requests and is considered as an external entity of the system. The communication component (*ComComp*) can only forward messages from one entity to the other. The access component (*AccComp*), in addition to forwarding messages, owns resources (*Resource*) and responds to requests by accessing these resources.

During experimentations, multiple formats for the message were used. We found that the most convenient method is to separate message parts depending on the security hypothesis of these parts.

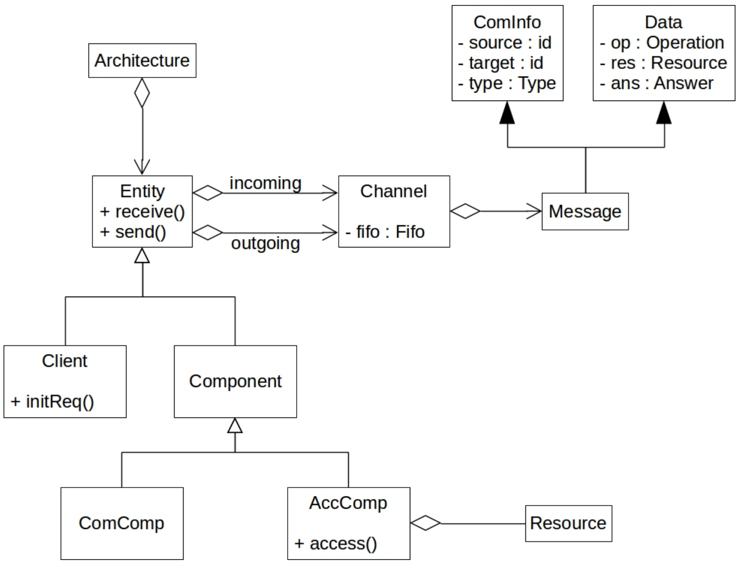


Figure 1. Abstract model

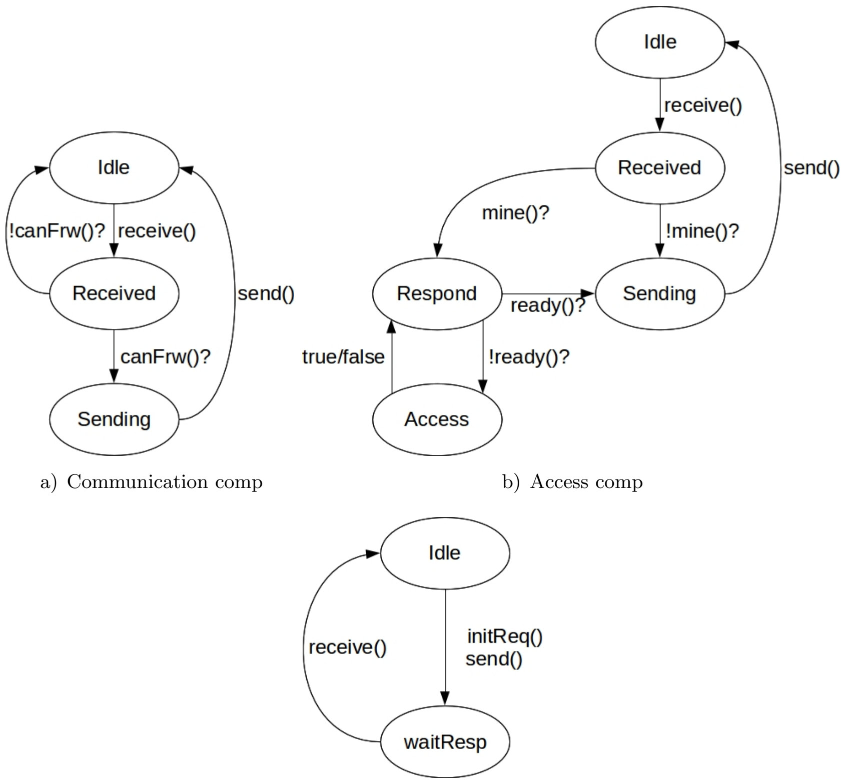
We consider the *comInfo* to have integrity, meaning if it is modified by an entity x, the source of the message becomes x. However, this part is not confidential and any other entity can read its contents so it can forward the message if needed. In addition for integrity, the *data* part is considered to have confidentiality, meaning that only the target of the message can read this part.

The message is therefore formatted as follows:

- *comInfo* containing information about the communication:
  - *source*: the source of the message,
  - *target*: the target of the message,
  - *type*: the type of the message, either a request *REQ* or a response *RESP*.
- *data* containing the details of the request or the response:
  - *op*: the operation concerned by the message (*READ* or *WRITE*),
  - *res*: the resource concerned by the message,
  - *ans*: the answer, *ACK* or *NAK* for *RESP* and *null* for *REQ*.

Figure 2 demonstrates the automata of a communication component, an access component, and a client, respectively. A communication component receives a message and goes to *Received* state, depending on whether it can forward the message or not, it either goes back to *Idle*, or it goes to *Sending*. An access component behaves differently after the *Received* state, if the target of the message is the component itself, it goes to *Respond*, else, it goes to *Sending* to forward the message. From

Respond, the access is accomplished by passing through *Access* and *Respond* again, finally, when the response is ready, the *Sending* state is visited to send the response. In the case of the client, it can initiate a request, and send this request, then it waits for a response at *waitResp*. When the response arrives, the client goes back to *Idle* state so it can send other requests.



a) Communication comp b) Access comp

c) Client

Figure 2. Automata of the interacting entities

In our approach, we assume that any component is physically robust, meaning that its behavior cannot be affected directly from the outside. The information passing from one part of the component to another is secure, it is not visible to other entities nor can they change it. The only way to affect a component is by sending a message to this component resulting in the component's behavior.

Our concept is to secure this behavior in a way that no matter what happens outside the components, the system behaves correctly, while no security properties are violated. Any integrated security pattern is also considered internal to the component. This means that any relation/communication/sharing between these patterns, and between these patterns and the component, is not visible to nor modifiable by any other entities.

Any implementation should respect this model, meaning that we can create more complicated entities, but the minimal structure should be respected. In addition, some behavior properties are used to limit possible problems and simplify the application of the patterns.

We start by explaining the required notations:

- $\mathcal{A}comps$, $\mathcal{C}comps$, and $\mathcal{C}lts$ are the sets of access components, communication components, and clients, respectively,
- $\mathcal{C}omps = \mathcal{A}comps \cup \mathcal{C}comps$ and $\mathcal{E}nts = \mathcal{C}omps \cup \mathcal{C}lts$,
- $received(e,\ m)$ is detected when entity e receives message m,
- $sent(e,\ m)$ is detected when e sends m,
- *any* is a joker used when the variable does not matter, for example, we write $received(e, any)$ detects the event where e received any message,
- $e.state$ is the current state of the entity e.

We formalize the properties shared by the different components as follows:

- Receive messages only when at *Idle* state.

$$\begin{aligned}&\textbf{prt\_ARCH\_1} : \forall c \in \mathcal{C}omps,\\ &\Box[received(c, any) \Rightarrow c.state = Idle].\end{aligned} \tag{1}$$

- Send messages only from *Sending* state.

$$\begin{aligned}&\textbf{prt\_ARCH\_2} : \forall c \in \mathcal{C}omps,\\ &\Box[sent(c, any) \Rightarrow c.state = Sending].\end{aligned} \tag{2}$$

- Never leave *Sending* state without sending a message.

$$\begin{aligned}&\textbf{prt\_ARCH\_3} : \forall c \in \mathcal{C}omps,\\ &\Box[c.state = Sending \Rightarrow \Diamond sent(c, any)].\end{aligned} \tag{3}$$

- To reduce the number of unnecessary configurations, some variables are given a null value (exp. $mess = MESS\_NULL$) when the component is at *Idle* state.

$$\begin{aligned}&\textbf{prt\_ARCH\_4} : \forall c \in \mathcal{C}omps,\\ &\Box[c.state = Idle \Rightarrow c.mess = MESS\_NULL].\end{aligned} \tag{4}$$

The formal description of the properties is close to implementation, since the included parameters and states are included in the minimal requirements for any application to our approach. However, when describing the security patterns in the following section, some formal properties are more abstract so they would consider different implementations.

4 SECURITY PATTERNS

The security pattern is a reusable solution to a recurring security problem. It is usually constructed by security specialists and used by developers who lack security knowledge. It provides a detailed guideline of how to implement the best found solution for a specific security problem.

Patterns should be considered as methodological tools to describe technical solutions related to security. They impose decisions that must be taken into consideration when designing architectures. They also facilitate communication between experts and non-experts.

We consider that the set of security measures to implement for an architecture are not, in general, managed at a single point. Due to the specificities of the patterns, the responsibility for managing the entire implemented security policy is shared between several patterns implemented on multiple components.

We can summarize the essential objective of each pattern as follows:

- *SAP* unifies the access points, instead of having to take security measures in different places, they are taken at the unified access point.
- *CHP* is dedicated to performing verifications and applying countermeasures based on a security policy.
- $AUTH$ implements a security policy based on access rights.
- $FWLL$ implements a security policy to restrict incoming and outgoing messages based on specified filters.

We divide these patterns into two categories: active patterns and passive patterns. *SAP* and *CHP* are the active patterns that should make decisions, calls, and verifications. $AUTH$ and $FWLL$ are the passive patterns which represent the forms that the security policies should take. Based on $AUTH$ and $FWLL$, *CHP* can verify the conformity of a request with the implemented security policy.

We consider two main objectives related to these patterns:

- Objective 1: Securing access to a component's resources:
  - *SAP* limits the access points to a component's resources to a single access point, and it checks the availability of requested resources.
  - *CHP*, solicited by *SAP*, verifies access rights and takes countermeasures.
  - $AUTH$ specifies the access rights.
- Objective 2: Securing access to a set of components:
  - *SAP* limits access points to an area to a single-point component, and it verifies the availability of the recipient entities of the messages.
  - *CHP* is called by *SAP* to verify that the message entering or exiting the area can pass (respects the security policy), if not, a countermeasure is taken.
  - $FWLL$ specifies the security policy used to filter messages.

4.1 Single Access Point (*SAP*)

The *SAP* pattern was introduced by Yoder and Barcalow [25]. It can be integrated into different types of implantation: a system, an application, a server, etc.

The goal of the *SAP* pattern is to unify the points on which an area can be accessed in order to improve the control and monitoring of entries.

Passing through a single access point is considered as a control, we therefore use the notation *controlled* on anything that passes through a single access point.

We use *SAP* for two types of control:

- Separating the architecture into areas with one access point between each two areas, and one access point to the external zone. In this case, *SAP* also verifies the availability of a target before forwarding a message. This control can be applied on communicating components with no resources to access. We call this SAP_C where C stands for communication.
- Separating the access to resources inside a component from the rest of the jobs handled in this component. In this case, at each access request, *SAP* also verifies if the targeted resource is available. This control can be applied on access components with resources to unify their access verifications. We call this SAP_A where A stands for access.

Since *SAP* is called each time, additional security patterns and security measures can later be patched to the *SAP* so they can handle other security requirements. After verifying the availability of the resource (or the component), if the verification fails, a negative response is prepared and sent. If the verification does not fail, either another security pattern(s) is/are called, or the request is fulfilled by accessing the resource (or forwarding the message).

Based on the *SAP* property classification proposed by Wassermann and Cheng [24], we have formalized some of these properties for different *SAP* applications. Notice that in this article we include some but not all of the properties.

We start by defining some required notations:

- $\mathcal{M}ess$ is the list of all possible messages.
- $\mathcal{R}es$ is the list of all resources.
- $\mathcal{O}pRes$ is the list of all possible operations on resources.
- $\mathcal{S}Ccomps$ is the list of all communication components using SAP_C.
- $\mathcal{S}Acomps$ is the list of all components using SAP_A.
- $\forall c_c \in \mathcal{S}Ccomps$, $c_c.comps$ is the list of all components considered inside the secure zone using c_c as a single access point.
- $\forall c_c \in \mathcal{S}Ccomps$, $\forall m \in \mathcal{M}ess$, $controlled(c_c, m)$ is true if the message m has already been controlled by c_c (passed through the single access point).
- $\forall c_a \in \mathcal{S}Acomps$, $c_a.res$ is the list of all resources owned by c_a.

- $\forall c_a \in \mathcal{S}Acomps$, $\forall e \in \mathcal{E}nts$, $\forall o \in \mathcal{O}pRes$, $accessed(c, e, o.oper, o.res)$ is true if e has accessed the resource $o.res$ in c_a with the operation $o.op$.
- $\forall c_a \in \mathcal{S}Acomps$, $\forall e \in \mathcal{E}nts$, $\forall o \in \mathcal{O}pRes$, $controlled(c, e, o)$ is true if c_a has already controlled the request o demanded by e.
- $m' = neg(c, Mess, m)$ is the error message corresponding to m, where c is the source of this error message where $m'.comInfo.target = m.comInfo.source$, $m'.comInfo.source = c$, and $m'.data.ans = NAK$.

We classify the properties depending on the *SAP* application, and the general type of the security property.

- Single Access Point of communication components (*SAP_C*):
  - Authenticity: All messages coming from outside an area protected by a component $c_c \in \mathcal{S}Ccomps$ should pass through this c_c before they go inside the area. Therefore, if a message is received by a component protected by c_c, and the source of the message is not protected by c_c, the message should have been controlled by c_c.

$$\begin{aligned}&\textbf{prt\_SAP\_C\_1.a}:\\&\forall m \in \mathcal{M}ess, \quad \forall c_c \in \mathcal{S}Ccomps, \quad \forall c \in c_c.comps,\\&\Box[received(c, m) \wedge m.source \notin c_c.comps\\&\ \Rightarrow controlled(c_c, m)].\end{aligned} \tag{5}$$

  - Any message received inside an area protected by component $c_c \in \mathcal{S}Ccomps$, without being controlled in advance by c_c, originated from inside the protected area.

$$\begin{aligned}&\textbf{prt\_SAP\_C\_1.b}:\\&\forall m \in \mathcal{M}ess, \quad \forall c_c \in \mathcal{S}Ccomps, \quad \forall c \in c_c.comps,\\&\Box[received(c, m) \wedge \neg controlled(c_c, m)\\&\ \Rightarrow m.source \in c_c.comps].\end{aligned} \tag{6}$$

  - Availability: When a message is received by a component $c_c \in \mathcal{S}Ccomps$, if it is coming from the outside to the inside of the protected zone (the source is not in the protected list), and the target is not available, a negative response is sent to the source of the message. We consider that a target is unavailable if it is not in the list of protected components.

$$\begin{aligned}&\textbf{prt\_SAP\_C\_2}:\\&\forall m \in \mathcal{M}ess, \quad \forall c_c \in \mathcal{S}Ccomps,\\&\Box[received(c_c, m) \wedge \{m.source, m.target\} \nsubseteq c_c.comps\\&\ \Rightarrow \Diamond sent(c_c, neg(c_c, m))].\end{aligned} \tag{7}$$

– Confidentiality: Messages exchanged between components inside a secure zone should never be read by outside entities.

$$\begin{aligned}&\textbf{prt\_SAP\_C\_3}:\\&\forall m \in \mathcal{M}ess, \quad \forall c_c \in \mathcal{S}Ccomps, \quad \forall e \in \mathcal{E}nts,\\&\Box[\mathit{received}(e,m) \wedge m.source \in c_c.comps\\&\;\wedge m.target \in c_c.comps \Rightarrow e \in c_c.comps].\end{aligned} \tag{8}$$

– Integrity: Messages exchanged between components inside a secure zone should never be modified by outside entities.

$$\begin{aligned}&\textbf{prt\_SAP\_C\_4}:\\&\forall m \in \mathcal{M}ess, \quad \forall c_c \in \mathcal{S}Ccomps, \quad \forall e \in \mathcal{E}nts,\\&\Box[\mathit{sent}(e,m) \wedge m.source \in c_c.comps \wedge m.target \in c_c.comps\\&\;\Rightarrow e \in c_c.comps].\end{aligned} \tag{9}$$

- Single Access Point of access components (*SAP_A*):

– Authenticity: Each access to a resource owned by component $c_a \in \mathcal{S}Acomps$ is controlled (passes through the single access point).

$$\begin{aligned}&\textbf{prt\_SAP\_A\_1}:\\&\forall c_a \in \mathcal{S}Acomps, \quad \forall e \in \mathcal{E}nt, \quad \forall o \in \mathcal{O}pRes,\\&\Box[\mathit{accessed}(c_a, e, o.oper, o.res)\\&\;\Rightarrow \mathit{controlled}(c_a, e, o)].\end{aligned} \tag{10}$$

– Availability: When a component $c_a \in \mathcal{S}Acomps$ controls an access, if the requested resource is not available, a negative response is sent (no matter what the operation is). The negative response is about the whole message and not only the access request, therefore, the received message is used to generate the negative response.

$$\begin{aligned}&\textbf{prt\_SAP\_A\_2}:\\&\forall c_a \in \mathcal{S}Acomps, \quad \forall m \in \mathcal{M}ess, \quad \forall r \in \mathcal{R}es,\\&\Box[\mathit{received}(c_a, m) \wedge \mathit{controlled}(c_a, m.comInfo.source,\\&(\mathit{any}, r)) \wedge r \notin c_a.res\\&\;\Rightarrow \Diamond \mathit{sent}(c_a, \mathit{neg}(c_a, m))].\end{aligned} \tag{11}$$

We notice that these properties do not mention what is authorized and what is not. As mentioned before, *SAP* only organizes and controls messages and access, without applying a security policy. In the case of other security patterns applied on the single access point, the properties of these patterns should consider the already applied pattern.

4.2 Check Point (*CHP*) Pattern

The main objective of *CHP* [25] is to enforce a security policy and activate appropriate countermeasures in the case of violations. While *CHP* can be found in some cases without *SAP*, in our work, we always apply these patterns together. Therefore, we can consider *SAP* in the formalization of *CHP* properties.

CHP does not contain the security policy, but enforces this policy. Therefore, in its formalization, we do not specify what the policy is, but only how to act depending on different situations regarding the policy.

For instance, if a certain policy is violated, the component can be temporarily shut, or can have a specific reaction varying according to what was violated and how. For simplicity, in this document, a countermeasure would be either to neglect a message or a request, or to send negative responses.

Whenever a single access point is visited, if *CHP* is included in the component, it should be visited before the continuation of the job. However, if there is an early refusal due to availability issues (unavailable target or resource), there is no need to call *CHP*.

In the same fashion as for *SAP*, we formalize some of the *CHP* properties extracted from the work of Wassermann and Cheng [24]. The following additional notations are required:

- $\mathcal{S}comps = \mathcal{SC}comps \cup \mathcal{SA}comps$ is the list of all secure components,
- $\forall c_s \in \mathcal{S}comps$, $c_s.policy$ is the security policy applied in c_s,
- $\mathcal{J}obs$ is the list of all possible jobs, which includes any resource access, and the forwarding of any message,
- $counter(p)$ is the countermeasure for violating the security policy p,
- $checked(c_s, job)$ is true if c_s has already checked if job is conform to the security policy applied in c_s,
- $accomplished(c_s, job)$ is true if c_s has accomplished job (the access is accomplished or the message is forwarded),
- $respects(p, job)$ is true if job respects the security policy p,
- $triggered(c_s, a)$ is true if c_s has triggered the action a.

The properties are:

- Availability: Each time *CHP* verifies a job that does not violate the policy, the job should continue (forwarding a message, or accessing a resource).

$$\begin{aligned}&\textbf{prt\_CHP\_1}:\\&\forall c_s \in \mathcal{S}comps, \quad \forall job \in \mathcal{J}obs,\\&\Box[checked(c_s, job) \wedge conform(c_s.policy, job)\\&\ \Rightarrow \Diamond accomplished(c_s, job)].\end{aligned} \tag{12}$$

- Availability: Each time *CHP* verifies a job which does violate the security policy, the appropriate countermeasure should be applied.

$$\begin{aligned}&\textbf{prt\_CHP\_2}:\\&\forall c_s \in \mathcal{S}comps, \quad \forall job \in \mathcal{J}obs,\\&\Box[checked(c_s, job) \wedge \neg conform(c_s.policy, job)\\&\Rightarrow \Diamond triggered(c_s, counter(c_s.policy))].\end{aligned} \tag{13}$$

- Authenticity: If a job is accomplished, then it must be verified, this includes the secured jobs only, such as forwarding messages or accessing resources.

$$\begin{aligned}&\textbf{prt\_CHP\_3}:\\&\forall c_s \in \mathcal{S}comps, \quad \forall job \in \mathcal{J}obs,\\&\Box[accomplished(c_s, job)\\&\Rightarrow checked(c_s, job) \wedge conform(c_s.policy, job)].\end{aligned} \tag{14}$$

4.3 Authorization Pattern ($AUTH$)

The authorization pattern ($AUTH$) was initially described by Fernandez et al. [14]. The main objective of $AUTH$ is to describe a security policy regarding secure access to objects. The concept is to specify protections on objects, then, create explicit access rights relations between some subjects and some objects so that the subjects have the privilege of accessing these objects.

In our work, the objects are the resources owned by the different components of type *AccComp*, and the subjects are any type of entity. Protections can be specified for an operation without the other (exp. resource r is protected for writing and unprotected for reading). Permissions can also be granted to an entity regarding specific operations or resources or both.

As mentioned before, *CHP* enforces the application of a security policy. In this case, the security policy ensured by *CHP* is an authorization pattern to ensure secure access to protected resources.

In order to apply the $AUTH$ policy on an access component, we use the syntax:

- $protect(c_a, op, r)$: Protect the resource r owned by component c_a for the operation *op*.
- $permit(e, c_a, op, r)$: Explicitly allow entity e to access the resource r owned by the component c_a using the operation *op*.

These functions can be used to specify the security policy which will later be combined with an insecure architecture generating a secure one. We can also use the wild-card *any* to generalize some of the rules. For example, $protect(c, any, r)$ means that resource r owned by component c should be protected for any type of access (read, write, execute, etc.).

Once protections and permissions are specified, we can use following predicates:

- $\mathit{protected}(c_a, op, r)$: which is true if the resource r owned by the access component c_a is protected for the operation op,
- $\mathit{permitted}(c_a, e, op, r)$: which is true if the entity e has explicit permission to access the resource r owned by the access component c_a using the operation op,
- $\mathit{allowed}(c_a, e, op, r)$: which is true if e should be allowed to access r owned by c_a using op. This is true if either the r is not protected for the operation op, or if there is a specific permission for e to have such access.

Notice that

$$\mathit{allowed}(c_a, e, op, r) = (\neg \mathit{protected}(c_a, op, r)) \vee (\mathit{permitted}(c_a, e, op, r)).$$

We are only interested in the essential property of $AUTH$ which specifies that each proceeded access is in fact allowed.

$$\begin{array}{l} \textbf{prt\_AUTH\_1}: \\ \forall c_a \in \mathcal{S}Acomps, \quad \forall e \in \mathcal{E}nts, \quad \forall o \in \mathcal{O}pRes, \\ \Box[\mathit{accessed}(c_a, e, o.oper, o.res) \\ \ \Rightarrow \mathit{allowed}(c_a, e, o.oper, o.res)]. \end{array} \tag{15}$$

4.4 Firewall Pattern ($FWLL$)

There are many descriptions for the Firewall pattern [21, 13, 7]. The main objective of $FWLL$ is to filter messages between different communication areas. It can function on the physical, transport, or application layer. At the physical layer, the firewall filter is applied to data packets transmitted over a network. Depending on information in these packets, some of them, or all of them, should be refused. At the transport layer, the filter collects packets until it has enough information to decide whether these packets are transferable or should be refused. Finally, at the application layer, $FWLL$ evaluates all messages against the associated rules concerning services. For example, a given service may not send messages to a certain entity.

In our work we consider a specific $FWLL$ approach which is most comparable to the application layer. In the same fashion as $AUTH$, the concept is to apply a specific security policy using $FWLL$. The policy decides on specific rules to refuse some messages depending on one or more variables in these messages. In addition, the policy can specify some exceptions for any of these rules.

As stated before, the *data* part of the message is confidential, and only the target can read it. This means that a firewall cannot base its filter on this part of the message. However, it can filter messages depending on their source, their target, and the type of the message.

In order to apply the $FWLL$ policy on a communication component, we use the following syntax:

- *protect*(c_c, *comps*): Applies the *FWLL* pattern to the communication component c_c. All components specified in the list *comps* are considered protected by this firewall, meaning that they are seen by the firewall as internal components. For instance, when *SAP_C* checks for target availability, this is the list it will check. In addition to having a list of inside components $c_c.comps$, c_c would also have $c_c.rules$ which are the rules for filtering messages based on their communication information *comInfo*.
- *addRule*(c_c, *comInfo*, *exps*): Adds a new filter rule to $c_c.rules$, the rule is based on *comInfo* stating the source, the target, and the type of the message. *exps* is a list of *comInfo* stating the exceptions to this rule.

In addition to using the wild-card *any* to generalize some of the rules and *null* to specify that there are no exceptions. For example, $addRule(c_c, (e_1, c_1, any), null)$ adds a rule to the policy of c_c stating that any message (of any type) with e_1 as its source and c_1 as its target is refused, with no exceptions.

We can check whether a message satisfies a rule or not using $satisfies(m, rule)$ which is true if $(m.comInfo \neq rule.comInfo) \vee rule.exps.contains(m.comInfo)$. Notice that the equality, as well as the *contains*, both understand the use of *any*. For example, the following statements are both true:

$$(e_1, c_1, req) = (e_1, c_1, any),$$

$$[(e_1, any, any)].contains((e_1, c_1, req)).$$

The most important property (regarding our work) of *FWLL* specifies that each forwarded message does not break any rules.

$$\begin{aligned}
&\mathbf{prt\_FWLL\_1}:\\
&\forall c_c \in \mathcal{SC}comps, \quad \forall, \forall m \in \mathcal{M}ess,\\
&\Box[received(c_c, m) \wedge sent(c_c, m)\\
&\ \Rightarrow \forall rule \in c_c.rules, satisfies(m, rule)].
\end{aligned} \tag{16}$$

5 SECURITY PATTERN COMPOSITION

Figure 3 demonstrates an abstraction of our combination approach. To generate a secure model, we specify the architecture elements along with the security policies. The security policies are translated into security requirements used to create the properties and assumptions of the system as well as the verification scenarios. The properties and assumptions are used to specify the pattern semantics (which security patterns are used and how).

Furthermore, the architecture, the pattern semantics and the scenarios are combined using our auto-combiner tool. The output contains the secure model, regular and attack scenarios, and observers coding the properties representing the security requirements of the architecture and the patterns. Finally, these outputs are used in a model-checker to verify that the model respects the properties in both regular and attack scenarios. The result of such verification is either a valid model or traces to help track violations.

For example, if we have the component c_1 in the architecture elements, and a security policy requiring to secure writing access to resource r_1 of c_1, the policy can be expressed using $protect(c_1, WRITE, r_1)$. When generating the secure model, we know that c_1 needs access protection, therefore we add the authorization pattern along with SAP_A and CHP. Since now c_1 has authorization pattern attached to it, each access to any of its resources should be verified. On the properties side, we add a property to verify that each access was either permitted because the resource is not protected for the specific operation (or an explicit permission is given to the source), or denied because the resource is protected (and no explicit permission is given). Finally, on the scenarios side, we add at least 2 different requests where one should be denied (lack of explicit permission) and one should be granted (explicit permission given). We can have other automatically generated scenarios (and some manual scenarios) to present different behaviors (exp. parallel requests) and make sure the properties are always respected.

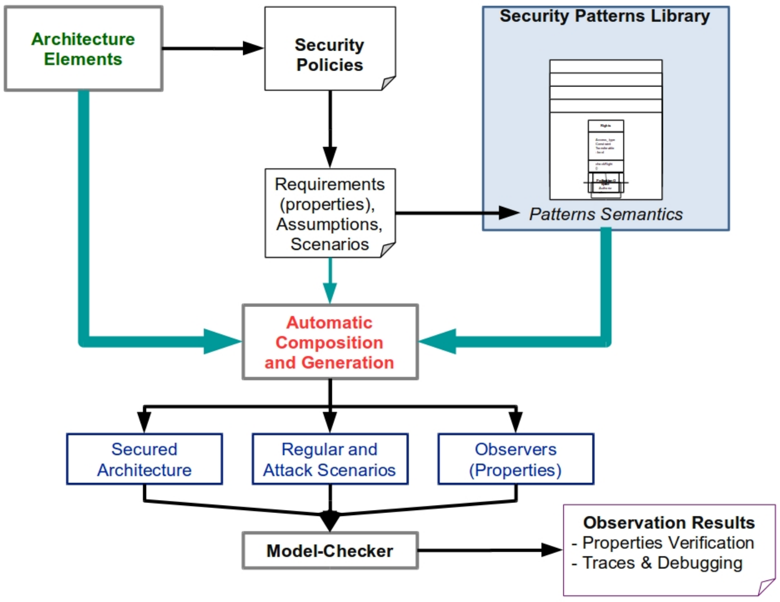

Architecture Elements
Security Policies
Security Patterns Library
Patterns Semantics
Requirements (properties), Assumptions, Scenarios
Automatic Composition and Generation
Secured Architecture
Regular and Attack Scenarios
Observers (Properties)
Model-Checker
Observation Results
- Properties Verification
- Traces & Debugging

Figure 3. Security pattern combination approach

The generated observers should consider the properties of the architecture, the properties regarding the security requirements, and the properties of the patterns used. For example, the architecture may have a specific behavior that needs to be respected, this should be translated into an observer to guarantee that the new behavior is correct. A security requirement regarding a protection of a resource also needs an observer to validate that the protection was indeed applied. Finally, the security patterns themselves have specific properties that should be respected.

In many cases, combining these different types of properties is a complicated task that can result in conflicts. This is not the same conflict we mentioned about multiple patterns used together: this conflict is about the requirements of different elements. For instance, the behavior of the architecture can be completely changed by applying a security policy. Consider that we have a model, and it should treat any message by forwarding it to a specific area. If the security policy limits access to this area, message treatment would now depend on the message itself.

Satisfying multiple requirements from different elements was a challenging part of our approach. However, to limit the automatic decision making, we consider the following order of property importance:

- The properties of the patterns are the most important, if these properties are not respected, the generated model can no longer be considered secure. Even if all other properties are respected or not, if the pattern is not functioning correctly, then any grantee given by the pattern can no longer be considered true.
- The security requirements are next, however, in implementation, we can combine these properties with the adjacent pattern properties. For example, the observer regarding a secure access requirement can be integrated in an $AUTH$ observer.
- Finally, the modified behavior of the model can have specific requirements that cannot be directly met due to the security measures. However, the new behavior is a combination between the original behavior, the patterns, and the requirements, which can be verified.

These are our own choices to resolve conflicts between properties. However, there can be other unresolved conflicts between properties which can be shown as properties violations which can be traced. To resolve these conflicts, either the security requirements need to be reduced or some services (model requirements) should be changed. The choice depends on each implementation, it also depends on the decision makers who decide which is more important, security or service. This question is not related to our work and is out of the scope of this paper. In our case, conflicts are resolved using the properties priorities stated above. Unresolvable conflicts did not appear during our experimentations.

6 VERIFICATION APPROACH

After specifying the architecture and the properties, we need to verify that these properties are respected in the secure architecture. In the case of complex systems, we cannot visually verify the code or the sequence diagram to ensure it respects its properties. A more suited solution is to use model checking tools to formally prove the correctness of a system. This approach relies on formal logic to obtain a proof that is absolute and undeniable.

To be able to formally represent and verify our model we need 3 essential elements (Figure 4).

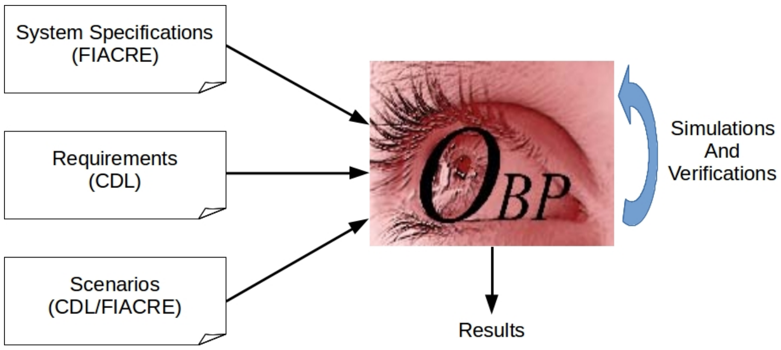


Figure 4. OBP explorer

We first need a language to model our SCADA system, the threat (attacker), and the security patterns. For this, we use the Intermediate Format for the Embedded Distributed Component Architectures (FIACRE) [4] which is a formally defined language that defines the behavior of a system considering time aspects. For the moment we are interested in the different possible behaviors of the system (with no consideration of time aspects).

We also need a way to formally present our properties that would ensure the fulfillment of the system requirements. We use the Context Description Language (CDL) [8] which, in addition to formalizing properties, can formalize scenarios to interact with the system. One of the main advantages of using CDL is the reduction in state explosion [9]. The scenarios can be either implemented using FIACRE or CDL, each implementation has its own advantages, in this article we have an example of each.

Finally, we need a model explorer to explore our model, and verify our properties. We use the Observer-Based Prover (OBP) explorer[1].

To use our model checking tools, we transform the different versions of our architecture (with or without each pattern) into a FIACRE code. We transform our scenarios and properties into CDL scenarios and CDL observers.

CDL uses a specific syntax to express properties which can be implemented using three different methods. While multiple methods can be used for the same purpose, each is more suitable for a specific situation:

[1] OBP documentation, tutorial and available tool: `http://www.obpcdl.org`

Invariant: Suitable in cases where only one configuration is concerned at a time. Exp. $pre_1 \Rightarrow pre_2$ means that at any configuration, if pre_1 is *true* then pre_2 should also be *true*.

Automaton: Suitable in cases where multiple configurations are concerned, it can detect events in the transitions from one configuration to another. Exp. $eve_1 \Rightarrow pre_2 \wedge eve_2$ means that if in a transition eve_1 happens, then pre_2 is *true* before the transition, and eve_2 happens in the same transition.

Liveness: Suitable when something leads to another. Exp. $pre_1 \Rightarrow \Diamond pre_2$, if pre_1 is *true*, pre_2 is eventually *true* in any following sequence.

For example, prt_CHP_3 uses the invariant method as follows:

$$\begin{aligned}
&predicate\ p_1\ is\ accomplished(c_i, job),\\
&predicate\ p_2\ is\ checked(c_i, job) \wedge conform(c_i.policy, job),\\
&assert\ not(p_1 \wedge \neg p_2).
\end{aligned} \tag{17}$$

This means that any configuration where p_1 is true and p_2 is false should be detected as a violation.

To observe prt_ARCH_1, we create the following automaton:

- $start \rightarrow e.Received(any) \rightarrow s1$ (1),
- $s1 \rightarrow e.state = Idle \rightarrow start$ (2),
- $s1 \rightarrow e.state \neq Idle \rightarrow reject$ (3).

The observer is at *start*, when e receives any message during a transition from one configuration to another, the observer goes to $s1$. During the same transition, the observer can make multiple shifts, from $s1$, if the state of e in the original configuration is *Idle*, the observer goes back to *start*. Since it has already detected the receiving event, it does not detect it again during this same transition.

It is important for the observer to go back to *start* in the case of correct behavior so it can detect the next time e receives a message. From $s1$, if the state of e in the original configuration is not *Idle*, the observer goes to *reject* so it can be easily traced. It stays in the *reject* state for the remainder of the sequence.

Figure 5 represents the property automaton, with p a priority to check one route before the other ($p1$ is tested before $p2$).

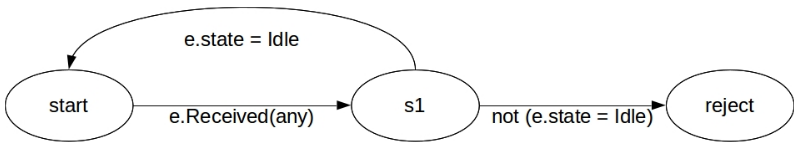


Figure 5. Initial property automaton

Finally, to observe *prt_SAP_C_2*, we use the liveness method as follows:

$$[\,]\ (|\ c_i@\mathit{Received} \land \{c_i.\mathit{mess.source}, c_i.\mathit{mess.target}\} \nsubseteq c_i.\mathit{comps}\ |$$

$$=><>|\ c_i@\mathit{Sending} \land c_i.\mathit{mess} = \mathit{neg}(c_i, c_i.\mathit{mess}))\ |).$$

If at some point c_i is at *Received*, and either the source of the message or its target is not in c_i*.comps*, then, eventually, c_i should be at *Sending* with a negative response. Since we already verify elsewhere that if c_i is at *Sending* it always sends a message, the combination leads to the property being correctly verified.

In the same fashion, we create observers for all of the other properties. Some of the more complicated properties require the use of more than one method at the same time.

7 EXPERIMENTATION

7.1 Process

In our experimentations, we tested multiple approaches on how to include the security patterns in the architecture. In this document we describe one of these approaches in which we use multiple states and transitions to apply the patterns. This modifies the automata (and the behavior) of both the communication component and the access component.

Figure 6 demonstrates the automaton of the secure communication component. From the *Received* state, in addition to verifying whether the message can be forwarded or not (basic verifications), we also verify if it is signed correctly. If this is the case, we go to *SAP* state which is responsible for testing the availability of the target. If the target is not available, a negative response is prepared and sent. If the target is available, *CHP* is called to verify the conformity of the message with the security policy. If the message conforms, it is forwarded, if not, *TrigAct* is called to make sure the correct countermeasure is applied, which is, in our case, sending a negative response.

Figure 7 demonstrates the automaton of the secure access component. From the *Received* state, in addition to verifying whether the target is the component itself, we also verify if the message is signed correctly. If this is the case, we go to *Respond* state which is responsible for preparing the correct response. In order to prepare the response, the resource needs to be accessed, and *SAP* is called to verify the availability of the resource. If it is not available, a negative response is prepared and sent. If the resource is available, *CHP* is called to test the conformity of the access demand with the security policy. If the demand conforms, the access is correctly achieved and a positive response is sent. If not, *TrigAct* is called to prepare a negative response as our chosen countermeasure.

The properties and scenarios can also be generated automatically. Each property generates an observer for each instance where it is necessary. For example, if we

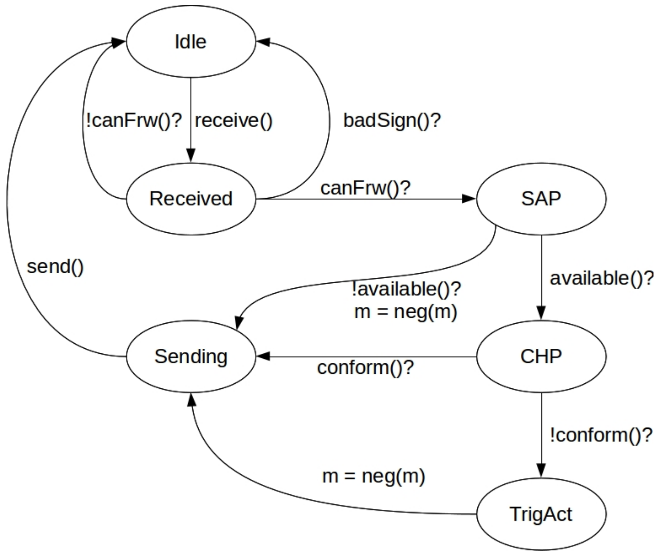


Figure 6. Secure communication component

have two access components c_a_1 and c_a_2, applying $prt\_SAP\_A\_1$ generates two observers $prt\_SAP\_A\_1\_c_a\_1$ and $prt\_SAP\_A\_1\_c_a\_2$.

This is applied differently depending on the property itself. For example, consider that the components above are both in an area secured by c_c_1. Applying the property $prt\_SAP\_C\_1.a$ creates an observer on c_c_1 itself to detect whether or not this happens. However, it should also verify the reaction of the components in the secure zone.

This means that the secure components should verify the signature correctly, which is observed using $prt\_SAP\_C\_1.a\_c_a\_1$ and $prt\_SAP\_C\_1.a\_c_a\_1$.

In this document we are more interested in validating our approach than verifying the generated models. Therefore, we use manually generated scenarios designed to verify certain aspects of the approach as well as to measure the complexity of the generated model.

We have multiple options to vary the scenarios:

- *Option_1*: Whether or not to have a stable system with an initial behavior. Which means, the system, without any external stimulation, already exchanges messages.
- *Option_2*: Whether or not to have normal interactions coming from the environment ($CLT1$ and $CLT2$ can send requests).
- *Option_3*: Whether or not to have attacking interactions coming from the environment (simply, an entity that is not recognized by the system, exp. $ATT1$).

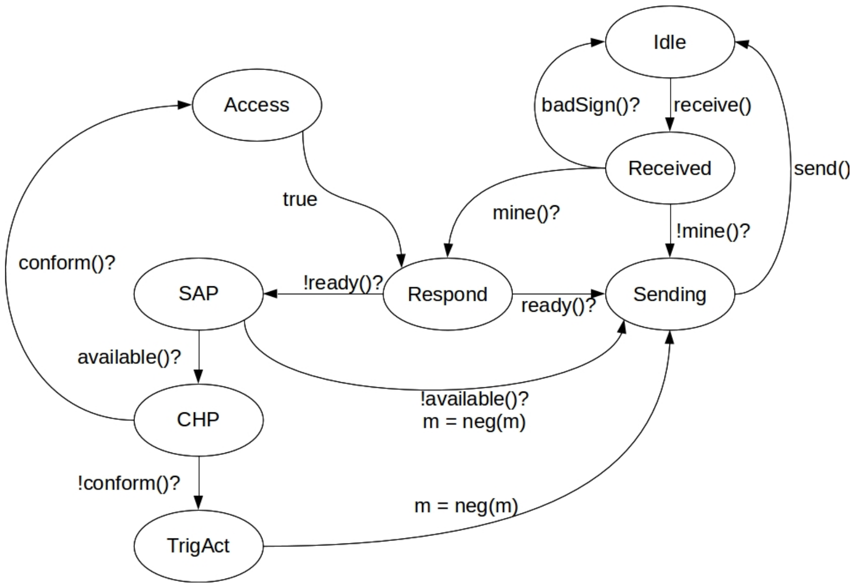


Figure 7. Secure access component

- *Option_4*: Whether or not to have attacking interactions from inside the system, we consider an unidentified entity (exp. $ATT2$) sending requests directly to $LC1$ without passing through GC and *Network*.

We can apply these options together. Consider that we have a system with inter-communications (*Option_1*), this would result in multiple possible configurations. We call this a stable system since each configuration can be reached from any other configuration. At any point (no matter which configuration the system is at), a client may send a valid request (*Option_2*) to which the response should be positive. An unidentified entity can also send a request (*Option_3*) which should result in a negative response. Finally, a request (*Option_4*) can be sent directly to an internal component (a component in a secure area), and since such requests were not signed by the *Firewall*, they should be ignored.

These scenarios are sufficient to verify all the different properties of the architecture, the security policy, and the patterns. However, the resulting configurations are too complicated for visual verification. Therefore, we start by simulating only one request using one of the options above, we alternate between requests and options to visually verify that the automatically generated model behaves as expected and the properties are verified correctly. Additionally, we add mistakes to the model to make sure the properties can detect these mistakes. Finally we can apply the full verification which can conclude that the secure model respects the properties.

7.2 Case Study

Consider we want to secure the model in Figure 8, it has the following components:

- GCS_1 and GCS_2: Access controllers which can access resources as well as forward messages. For the sake of our example, their access capabilities will not be necessary since there are no requests targeting them.
- NET_1 and NET_2: Communication controllers which can only forward requests. Both have a firewall when security is applied. NET_1 would protect every other component other than GCS_1, and NET_2 would protect every component of type device DEV_i.
- Four PLC_i each responsible for the corresponding device DEV_i. They ensure indirect access, each request to a PLC_i is treated at PLC_i before being forwarded to the device DEV_i. All PLC_i have secure access when security is applied.
- Four DEV_i which respond to the indirect access requests. Note that they do not apply security measures other then verifying messages signatures. Access rights verifications are run at PLC_i level.

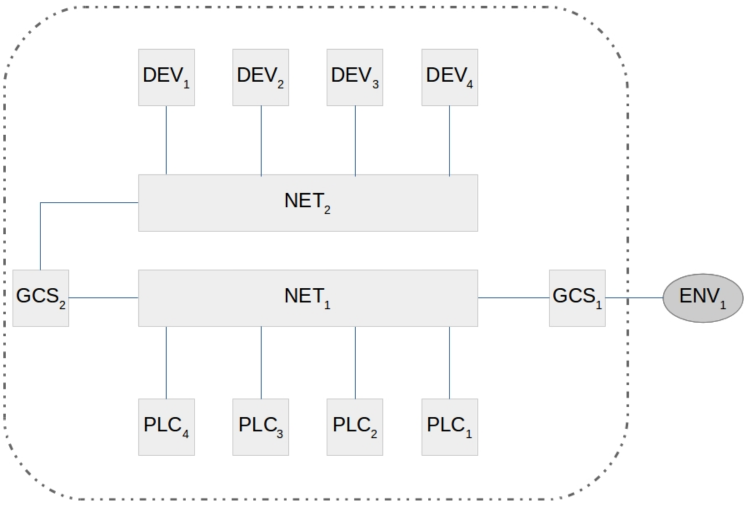


Figure 8. Simple SCADA model

For example, a request from CLT_i to PLC_i would go through GCS_1, NET_1, PLC_i, NET_1, GCS_2, NET_2, and DEV_i. The response would take the reverse route exactly. However, the request can be refused at NET_1 and NET_2 in the case of target unavailability or filter violation, or at PLC_i in the case of resource unavailability or access rights violation.

We apply a security policy on this model regarding message filtering and resource access. The policy details are:

- $\mathit{protect}(\mathit{GCS}_i, \mathit{any}, \mathit{any})$: Protect all resources of any GCS for any operation (we use i to apply the rule for any component of this type).
- $\mathit{protect}(\mathit{PLC}_i, \mathit{any}, \mathit{any})$: Protect all resources of any PLC for any operation.
- $\mathit{protect}(\mathit{NET}_1, \{\mathit{PLC}_i, \mathit{GCS}_2, \mathit{NET}_2, \mathit{DEV}_i\})$: Apply a firewall on NET_1 protecting all *PLCs* and *Devs*, and NET_2 and GCS_2.
- $\mathit{protect}(\mathit{NET}_2, \{\mathit{DEV}_i\})$: Apply a firewall on NET_2 protecting all *Devs.*
- $\mathit{addRule}(\mathit{NET}_2, (\mathit{any}, \{\mathit{DEV}_i\}, \mathit{any}), \{(\{\mathit{PLC}_i\}, \mathit{any}, \mathit{any})\})$: Add a filter rule to NET_2 refusing any message targeting DEV unless the source of this message is a *PLC.*

We create multiple scenarios which we simulate on both the secure and the insecure model. For verification purposes, we consider four different types of actors included in these scenarios:

- CLT_i: A legit client to which the response should be positive.
- SPM_i: A known non-legit entity whose requests should be stoppable at Firewalls. This is implemented using $\mathit{addRule}(\mathit{NET}_i, (\{\mathit{SPM}_i\}, \mathit{any}, \mathit{any}), \{\})$
- ATT_i: An unknown non-legit entity whose requests may pass through the Firewalls but has no access rights resulting in negative responses directly from the PLC_i (without forwarding the request to the device DEV_i).
- PEN_i: A penetration entity which would send requests directly from inside the system, either to PLC_i without passing through NET_1 or to DEV_i without passing through NET_2.

In the case of the insecure model, apart from PEN_i, we expect that the behavior of the other actor types is always the same. Since there is no way of filtering messages or applying access rights, these actor types are all the same to the system and their messages always result in positive responses. PEN_i requests would also have positive responses, however the route of the request is shorter since it is directly inserted inside the system.

When security is applied, the behavior is expected to differ for each actor type. CLT_i should stay the same since it is legit, the route of the request is the same, however the number of resulting configurations is expected to increase because of the states and transitions related to security. SPM_i requests are directly refused at NET_1 resulting in a negative response caused by the applied filter. ATT_i routes are longer than SPM_i and shorter than CLT_i since they are refused at PLC_i level. Finally, PEN_i messages are refused directly by the receiving component without calling *SAP*, *CHP*, nor *TrigAct*, since they are not signed and this is verified at *Received* state. This results in the shortest route for the PEN_i requests.

7.3 Property Verification

First, we investigated the insecure and the secure models by observing their behavior when stimulated by different possible actors, separated, or combined. During these observations we confirmed our previous expectation about both models. We also confirmed that the properties of the patterns, the security policy, and the initial model are all respected in the secure model. This is true not only for the properties mentioned in this paper but also for all the properties of the patterns.

As explained before, the properties are applied for each component concerned. For simplicity, we do not go into the details of each predicate and event used, nor how they are written: we are more interested in the format of the properties. As explained before, the properties are applied for some or all of the components depending on the property, some are only in the secure model while others are used in both models. In the following, we present some of the properties verified using our approach. Each property is written for only one component, other applications undergo slight modifications such as the name of the component.

- *prt_ARCH_1*: Applied on each and every component in both the insecure and the secure models. The following is its application on GCS_1:

$$\begin{array}{l} \textit{property prt\_ARCH\_1\_GCS}_1 \textit{ is} \\ \textit{start} \; -- \; // \; \textit{recv\_GCS}_1\textit{\_ANY} \; / -> \; \textit{wait}; \\ \textit{wait} \; -- \; / \; \textit{pre\_GCS}_1\textit{\_Idle} \; // -> \; \textit{start}; \\ \textit{wait} \; -- \; / \; \textit{not pre\_GCS1\_Idle} \; // -> \; \textit{reject.} \end{array} \tag{18}$$

  $recv\_GCS_1\_ANY$ is detected when GCS_1 receives any message. At this point, if GCS_1 was at *Idle* ($pre\_GCS_1\_Idle$), the property is respected and the observer goes to *start* to wait for the next time GCS_1 receives a message. If not, the property is rejected and the configuration where this violation happened can be easily traced. Notice that, at any configuration, the observer is either at *start* or *reject*, this is important so that the property is verified each time GCS_1 receives a message.

- $prt\_SAP\_C\_2\_NET_2$ (Figure 9): Applied on any component applying SAP_C which are NET_1 and NET_2. For example, its application on NET_2 is as follows:

$$\begin{array}{l} \textit{property prt\_SAP\_C\_2\_NET}_2 \textit{ is} \\ \textit{start} \; --// \; \textit{eve\_NET}_2\textit{\_SAP\_true} \; / -> \; \textit{at\_Sap}; \\ \textit{at\_Sap} \; --/ \; \textit{not pre\_NET}_2\textit{\_Available} \; /\textit{eve\_NET}_2\textit{\_Sending\_true} \; / -> \; \textit{ver}; \\ \textit{at\_Sap} \; --/ \; \textit{not pre\_NET}_2\textit{\_Available} \; /\textit{eve\_NET}_2\textit{\_SAP\_false} \; / -> \; \textit{reject}; \\ \textit{at\_Sap} \; --/ \; \textit{pre\_NET}_2\textit{\_Available} \; /\textit{eve\_NET}_2\textit{\_SAP\_false} \; / -> \; \textit{start}; \\ \textit{ver} \; --/ \; \textit{pre\_NET}_2\textit{\_NAK} \; /\textit{send\_NET}_2\textit{\_ANY} \; / -> \; \textit{start}; \\ \textit{ver} \; --/ \; \textit{not pre\_NET}_2\textit{\_NAK} \; /\textit{send\_NET}_2\textit{\_ANY} \; / -> \; \textit{reject} \end{array} \tag{19}$$

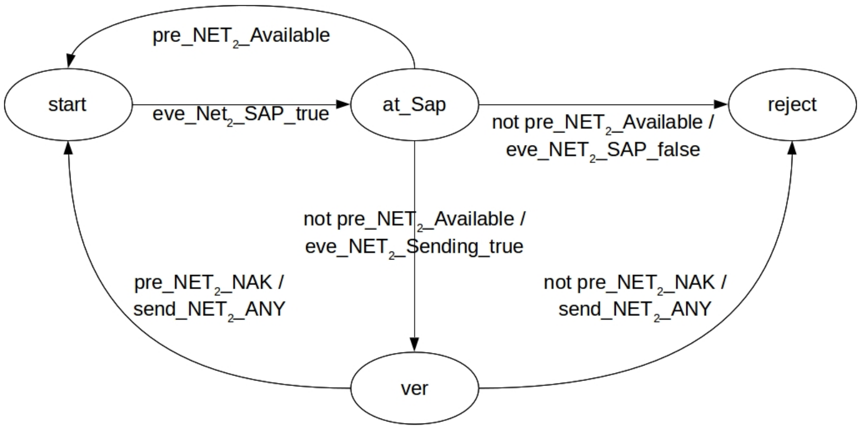


Figure 9. Application of Property_SAP_C_2 on Net2

This means that, if NET_2 goes to *SAP* state ($eve\_NET2\_SAP\_true$), the observer goes to *at_Sap*. If the target is not available (*not* $pre\_NET2\_Available$) and NET_2 goes to *Sending*, the observer goes to *ver*.

If the target is not available, and NET_2 changes its state (other than going to *Sending*), the property is rejected. However, if the target is available, the observer goes back to *start*. This is because this property is only interested in cases where the target is not available.

From *ver*, if NET_2 has a *NAK* message and sends it, the property is respected, and the observer goes back to start. However, if the message is not a *NAK* message, the observer goes to *reject* the moment this message is sent (not before).

Notice that in this example, we can have configurations where the observer is not at *start* nor *reject*, however it would still work each time *SAP* state is visited. This is because, from *SAP*, if *NET_2* changes its state, the observer goes to *start*, *reject*, or *ver* if NET_2 went to *Sending*.

Moreover, from *Sending*, we already verify that the component cannot leave without sending a message. NET_2 either has $pre\_NET_2\_NAK$ true or false, so it either goes to *start* or *reject*. In short, at any point where NET_2 can go to *SAP*, the observer is either at *start* or *reject*.

- prt_CHP_2: Applied on any component applying *CHP* which is every component apart from DEV_i devices. For example, on PLC_1:

$$\begin{aligned} &\textit{property } prt\_CHP\_2\_PLC_1 \textit{ is}\\ &[](\mid pre\_PLC_1\_CHP \wedge \neg pre\_PLC_1\_conform\\ &=><>\mid pre\_PLC_1\_TrigAct)\mid) \end{aligned} \tag{20}$$

The property verifies that if, at any configuration, PLC_1 is at *CHP* and the request is not conform to the policy, eventually PLC_1 should go to *TrigAct*.

- prt_ARCH_4: Applied on any component, its application on DEV_1:

$$\begin{aligned}&\textit{property } prt\_ARCH\_4\_DEV_1 \textit{ is}\\&\textit{assert not}(pre\_DEV_1\_Idle \textit{ and } \neg DEV_1.mess == MESS\_NULL).\end{aligned} \tag{21}$$

Table 1 demonstrates for each property on which component this property is applied. The application depends on the situation of the component, its characteristics, and its type. In this table, when the component type is placed instead of the component, it means the property is applicable on each component of this type.

For example, $prt\_SAP\_C\_2$ is applied on any component of type *NET*, which are NET_1 and NET_2. Also, ENV_a means all entities in the environment.

Property	**Type**	**Applied on**	**Components**
prt_ARCH_1	safety	all components	GCS, NET, PLC, DEV
prt_ARCH_2	safety	all components	GCS, NET, PLC, DEV
prt_ARCH_3	safety	all components	GCS, NET, PLC, DEV
prt_ARCH_4	safety	all components	GCS, NET, PLC, DEV
$prt\_SAP\_C\_1.a$	authenticity	comps behind a firewall	GCS_2, NET_2, PLC, DEV
$prt\_SAP\_C\_1.b$	authenticity	comps behind a firewall	GCS_2, NET_2, PLC, DEV
$prt\_SAP\_C\_2$	availability	secure communication comps	NET
$prt\_SAP\_C\_3$	confidentiality	all entities	GCS, NET, PLC, DEV, ENV
$prt\_SAP\_C\_4$	integrity	all entities	GCS, NET, PLC, DEV, ENV
$prt\_SAP\_A\_1$	authenticity	secure access comps	GCS, PLC
$prt\_SAP\_A\_2$	availability	secure access comps	GCS, PLC
prt_CHP_1	availability	secure comps	GCS, NET, PLC
prt_CHP_2	availability	secure comps	GCS, NET, PLC
prt_CHP_3	authenticity	secure comps	GCS, NET, PLC
prt_AUTH_1	authenticity	secure access comps	GCS, PLC
prt_FWLL_1	authenticity	secure communication comps	NET

Table 1. Properties application

7.4 Complexity Measurements

We use dedicated scenarios based on the simple actors of type CLT_i. Here we are interested in analyzing the complexity of our approach regarding the resulting configurations. Therefore, scenarios should be normalized in a way that each request takes (by itself) exactly the same amount of transitions to be fulfilled. When combined, multiple requests from the same actor or multiple actors would have a great impact on the resulting configurations. The main objective here is to confirm that the impact of applying the security patterns is limited and predictable.

For the complexity measurements, we do not use the other actor types since we have already confirmed that, when used on the secure model, their routes are less important in terms of complexity. We have multiple scenarios noted A_i_j where i is the number of actors (CLT_i) included in the scenario and j is the number of messages sent by this actor. Multiple actors can send requests in parallel, however, each actor waits for a response before sending its following request. We have a maximum of four requests per actor, which are $(READ, res_1)$, $(WRITE, res_1)$, $(READ, res_2)$, and $(WRITE, res_2)$, We also have a maximum of four actors each sending requests to the corresponding PLC_i. Each received message cannot cause more than one message sent (either forward the message or send a response or ignore). Therefore, we know in advance that it is impossible for our model to have more than 4 messages in its channels all at once. We fixed the size of all channels to 5 and observed that no channel was ever full, this helped in the normalization between the scenarios so they would not be affected by the size of the channels.

Table 2 contains the number of configurations, the number of transitions, and the depth (D), for each scenario and for both secure and insecure models.

Figure 10 demonstrates the ratios between secure and insecure models in terms of number of configurations, number of transitions, and depth. We compute these ratios for different scenarios with different number of actors and number of messages per actor. The depth ratio is completely stable. This is because the added number of transitions to respond to a request is stable for legit requests. Both configuration and transition ratios are stable in the case of one actor. These ratios increase by around 0.3 points each time we add a new actor. However, their increase due to messages per actor increase is much less significant.

These results show that using the security patterns, the complexity of our model increases but in a stable and predictable fashion. Note that, in cases where both legit requests and non-legit requests are used, the complexity may decrease since most non-legit requests are treated with less transitions in the case of the secure model than the insecure one.

8 CONCLUSION

Current SCADA systems have a shortage in security measures, while the interest of attackers in these systems is constantly increasing. We do not limit our work to the SCADA domain, however, we are interested in considering it in our simulations since it represents a challenge when it comes to information security. The reflections developed in this work are generic and could be adapted to other types of architectures.

	Insecure Model			Secure Model		
Scn	**nb. confs**	**nb. trans**	**D**	**nb. confs**	**nb. trans**	**D**
*A*_1_1	49	48	48	63	62	62
*A*_1_2	97	96	96	125	124	124
*A*_1_3	145	144	144	187	186	186
*A*_1_4	193	192	192	249	248	248
*A*_2_1	2 344	4 415	96	3 686	6 977	124
*A*_2_2	11 207	21 424	192	17 755	34 028	248
*A*_2_3	26 590	51 027	288	42 208	81 153	372
*A*_2_4	48 493	93 224	384	77 045	148 352	496
*A*_3_1	110 137	299 913	144	204 621	560 051	186
*A*_3_2	1 394 467	3 871 044	288	2 623 843	7 301 556	372
*A*_3_3	5 396 320	15 039 747	432	10 183 726	28 435 959	558
*A*_3_4	13 605 189	37 982 760	576	25 707 157	71 886 920	744
*A*_4_1	5 114 765	17 939 618	192	10 953 437	38 624 884	248
*A*_4_2	180 143 913	646 742 856	384	392 578 705	1 412 820 112	496
*A*_4_3	1 150 191 955	4 145 224 977	576	2 515 317 972	9 082 222 251	744

Table 2. Complexity measurements

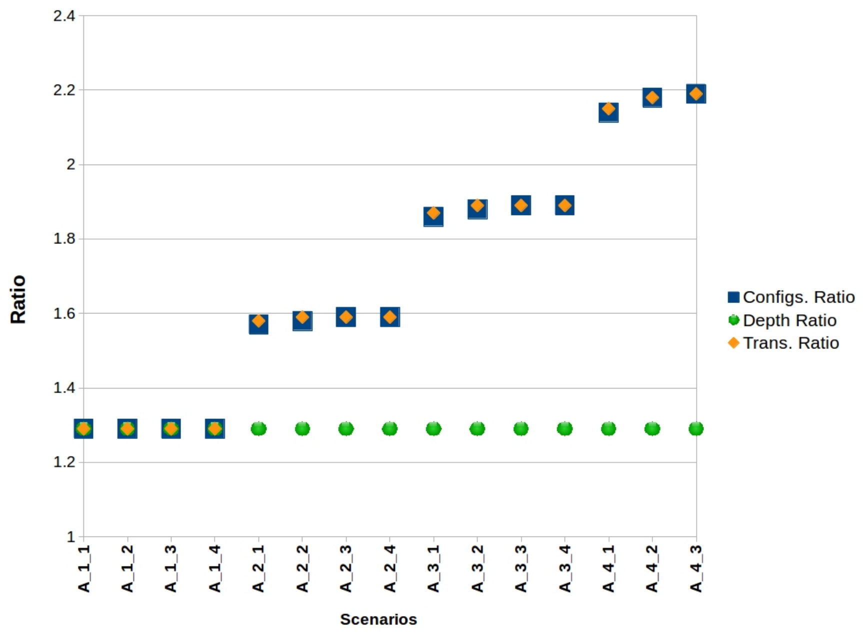


Figure 10. Ratios between secure and insecure models

[12] proposes to use security patterns to strengthen the security of SCADA architectures. Security patterns allow us to implement current best found solutions to resolve specific security issues. Our goal was to provide rules to generate a secure model based on an insecure one, some security requirements, and some security patterns to satisfy these requirements. These rules can be used to automatically generate a secure model based on the insecure one.

We also looked into tools to formally verify validate the generated secure model using model checking techniques. This validation goes through the verification of the properties of the model, the properties of the security patterns, and the properties related to the security requirements.

Our experiments have strengthened our ability to drive, by this method and technique, a process of integration of the security patterns and formal validation of the generated model. But given the large number of properties and scenarios involved in this process, and to be taken into account in the verification process, the approach only makes sense if we are able to generate these sets of properties and scenarios.

During the experiments, we limited ourselves to four security patterns. But a similar approach (formalization, composition, verification) could be used for the integration of other patterns described in the literature. We also focused on sets of properties that could be extended for each pattern.

Currently we are working on improving the combination and generation rules. In our work, we opted for implementation choices which can be studied and optimized further. In addition, we are updating our library to include more patterns. Finally, we seek to include an attack library which, in addition to having generic attack scenarios, allows us to have scenarios representing specific attacks and threats.

Acknowledgement

Partially supported by the Research Grant of the French Government Defense Procurement and Technology Agency – Information Department (DGA-MI).

REFERENCES

[1] Alencar, P.—Cowan, D.—Dong, J.—Lucena, C.: A Pattern-Based Approach to Structural Design Composition. Proceedings of the Twenty-Third Annual International Computer Software and Applications Conference (COMPSAC '99), IEEE, 1999, pp. 160–165, doi: 10.1109/CMPSAC.1999.812694.

[2] Alencar, P. S. C.—Cowan, D. D.—Lucena, C. J. P.: A Formal Approach to Architectural Design Patterns. In: Gaudel, M. C., Woodcock, J. (Eds.): Industrial Benefit and Advances in Formal Methods (FME '96). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 1051, 1996, pp. 576–594, doi: 10.1007/3-540-60973-3_108.

[3] Avalle, M.—Pironti, A.—Sisto, R.: Formal Verification of Security Protocol Implementations: A Survey. Formal Aspects of Computing, Vol. 26, 2014, No. 1, pp. 99–123, doi: 10.1007/s00165-012-0269-9.

[4] Berthomieu, B.—Bodeveix, J.-P.—Farail, P.—Filali, M.—Garavel, H.—Gaufillet, P.—Lang, F.—Vernadat, F.: Fiacre: An Intermediate Language for Model Verification in the Topcased Environment. 4th European Congress ERTS Embedded Real Time Software (ERTS 2008), Toulouse, France, 2008, 8 pp.

[5] Byres, E.—Lowe, J.: The Myths and Facts Behind Cyber Security Risks for Industrial Control Systems. Proceedings of the VDE Kongress, Vol. 116, 2004, pp. 213–218.

[6] Clarke, E. M.—Grumberg, O.—Peled, D.: Model Checking. MIT Press, 1999.

[7] Delessy-Gassant, N.—Fernandez, E. B.—Rajput, S.—Larrondo-Petrie, M. M.: Patterns for Application Firewalls. Proceedings of the Conference on Pattern Languages of Programs (PLoP), 2004.

[8] Dhaussy, P.—Boniol, F.—Roger, J.-C.—Leroux, L.: Improving Model Checking with Context Modelling. Advances in Software Engineering, Vol. 2012, 2012, Art. No. 547157, 13 pp., doi: 10.1155/2012/547157.

[9] Dhaussy, P.—Roger, J.-C.—Boniol, F.: Reducing State Explosion with Context Modeling for Model-Checking. Proceedings of the 2011 IEEE 13th International Symposium on High-Assurance Systems Engineering (HASE '11), 2011, pp. 130–137, doi: 10.1109/HASE.2011.24.

[10] Dong, J.—Alencar, P. S. C.—Cowan, D. D.: Ensuring Structure and Behavior Correctness in Design Composition. Proceedings of the Seventh IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS 2000), 2000, pp. 279–287, doi: 10.1109/ECBS.2000.839887.

[11] Dong, J.—Peng, T.—Zhao, Y.: Model Checking Security Pattern Compositions. Proceedings of the Seventh International Conference on Quality Software (QSIC '07), IEEE, 2007, pp. 80–89, doi: 10.1109/QSIC.2007.4385483.

[12] Fernandez, E. B.—Larrondo-Petrie, M. M.: Designing Secure SCADA Systems Using Security Patterns. Proceedings of the 2010 43rd Hawaii International Conference on System Sciences (HICSS '10), IEEE, 2010, pp. 1–8, doi: 10.1109/HICSS.2010.139.

[13] Fernandez, E. B.—Larrondo-Petrie, M. M.—Seliya, N.—Delessy-Gassant, N.—Herzberg, A.: A Pattern Language for Firewalls. 2003.

[14] Fernandez, E. B.—Pan, R.: A Pattern Language for Security Models. Proceedings of the PLoP Conference, Vol. 1, 2001.

[15] Fovino, I. N.—Coletta, A.—Carcano, A.—Masera, M.: Critical State-Based Filtering System for Securing SCADA Network Protocols. IEEE Transactions on Industrial Electronics, Vol. 59, 2012, No. 10, pp. 3943–3950, doi: 10.1109/TIE.2011.2181132.

[16] Fovino, I. N.—Coletta, A.—Masera, M.: Taxonomy of Security Solutions for the SCADA Sector. Project ESCORTS Deliverable, Vol. 2, 2010.

[17] Hafiz, M.—Adamczyk, P.—Johnson, R. E.: Growing a Pattern Language (for Security). Proceedings of the ACM International Symposium on New Ideas, New

Paradigms, and Reflections on Programming and Software (Onward! 2012), 2012, pp. 139–158, doi: 10.1145/2384592.2384607.

[18] Igure, V. M.—Laughter, S. A.—Williams, R. D.: Security Issues in SCADA Networks. Computers and Security, Vol. 25, 2006, No. 7, pp. 498–506, doi: 10.1016/j.cose.2006.03.001.

[19] Krutz, R. L.: Securing SCADA Systems. John Wiley and Sons, 2005.

[20] Viega, J.—McGraw, G.: Building Secure Software: How to Avoid Security Problems the Right Way. 1st Edition. Addison-Wesley Professional, 2001.

[21] Schumacher, M.—Fernandez-Buglioni, E.—Hybertson, D.—Buschmann, F.—Sommerlad, P.: Security Patterns: Integrating Security and Systems Engineering. John Wiley and Sons, 2013.

[22] Taibi, T.—Ngo, D. C. L.: Formal Specification of Design Pattern Combination Using BPSL. Information and Software Technology, Vol. 45, 2003, No. 3, pp. 157–170, doi: 10.1016/S0950-5849(02)000195-7.

[23] Wang, Y.: sSCADA: Securing SCADA Infrastructure Communications. International Journal of Communication Networks and Distributed Systems, Vol. 6, 2011, No. 1, pp. 59–78, doi: 10.1504/IJCNDS.2011.037328.

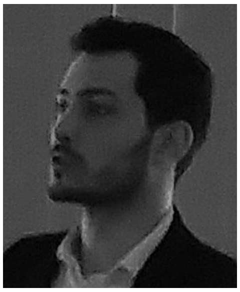
[24] Wassermann, R.—Cheng, B. H.: Security Patterns. Proceedings of the PLoP Conference, Michigan State University, 2003, Citeseer.

[25] Yoder, J.—Barcalow, J.: Architectural Patterns for Enabling Application Security. PLoP '97 Conference, Urbana, Vol. 51, 1998, Art. No. 61801.

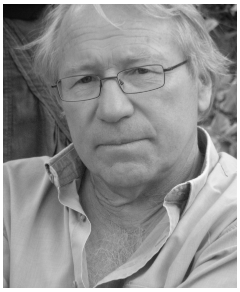
[26] Yoshioka, N.—Washizaki, H.—Maruyama, K.: A Survey on Security Patterns. Progress in Informatics, Vol. 5, 2008, pp. 35–47, doi: 10.2201/NiiPi.2008.5.5.

[27] Zhu, B.—Joseph, A.—Sastry, S.: A Taxonomy of Cyber Attacks on SCADA Systems. Proceedings of the 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing (iThings/CPSCom 2011), IEEE, 2011, pp. 380–388, doi: 10.1109/iThings/CPSCom.2011.34.

[28] Zhu, B.—Sastry, S.: SCADA-Specific Intrusion Detection/Prevention Systems: A Survey and Taxonomy. Proceedings of the 1st Workshop on Secure Control Systems (SCS), 2010.



Fadi Obeid is consultant in cyber security, working on different cyber security subjects, for multiple clients. He studied cyber security in the University of Limoges, France, and obtained his master's degree in 2013. He started his Ph.D. at ENSTA Bretagne in 2014 where he worked on model checking security patterns implementations. In parallel to his Ph.D. at ENSTA Bretagne, he created a new encryption protocol dedicated to the low vocabulary of SCADA and IoT communications. After finishing his Ph.D. in 2018, he started working as Consultant for Alter Solutions. He has a polyvalent profile, including model checking, cryptography, side channel analysis, functional security.



Philippe Dhaussy is Professor at CNRS Lab-STICC within ENSTA Bretagne. His expertise and his research interests include model-driven software engineering, formal validation for real time systems and embedded software design. He has an engineer degree in computer science from ISEN (French Institute of Electronics and Computer Science) in 1978 and received his Ph.D. in 1994 at Telecom Bretagne (France) and his HDR in 2014. From 1980 to 1991, he had been software engineer and technical coordinator in consulting companies (Atlantide group), mainly in real-time system developments. He joined ENSTA-Bretagne in 1996 as Professor. He has over 100 publications in the areas of software engineering and computer science. He co-supervised ten Ph.D. students and he was and still is involved in several research projects as a work package coordinator.

Computing and Informatics, Vol. 38, 2019, 1181–1209, doi: 10.31577/cai_2019_5_1181

COALGEBRAIC OPERATIONAL SEMANTICS FOR AN IMPERATIVE LANGUAGE

William Steingartner, Valerie Novitzka

Faculty of Electrical Engineering and Informatics
Technical University of Košice
Letná 9, 042 00 Košice, Slovakia
e-mail: {william.steingartner, valerie.novitzka}@tuke.sk

Wolfgang Schreiner

Research Institute for Symbolic Computation
Johannes Kepler University
Altenberger Straße 69, A-4040 Linz, Austria
e-mail: wolfgang.schreiner@risc.jku.at

Abstract. Operational semantics is a known and popular semantic method for describing the execution of programs in detail. The traditional definition of this method defines each step of a program as a transition relation. We present a new approach on how to define operational semantics as a coalgebra over a category of configurations. Our approach enables us to deal with a program that is written in a small but real imperative language containing also the common program constructs as input and output statements, and declarations. A coalgebra enables to define operational semantics in a uniform way and it describes the behavior of the programs. The state space of our coalgebra consists of the configurations modeling the actual states; the morphisms in a base category of the coalgebra are the functions defining particular steps during the program's executions. Polynomial endofunctor determines this type of systems. Another advantage of our approach is its easy implementation and graphical representation, which we illustrate on a simple program.

Keywords: Category theory, coalgebra, operational semantics, programming language

Mathematics Subject Classification 2010: 18C50, 16T99, 97P40

1 INTRODUCTION

Formal semantics belongs inherently to the definition of programming languages. It provides the meaning of the programs in an unambiguous way. There are several well-known semantic methods for defining the formal semantics of programs written in some programming language. The choice of a suitable semantic method depends on the information we desire. In this paper, we are interested in operational semantics of a simple imperative language *E-Jane* defined as a coalgebra over a category of configurations.

Operational semantics is the most popular semantic method. It was defined by Plotkin in [24]; and he discusses its motivation in [25]. This method describes not only the meaning of a program but also the execution details. Operational semantics can be considered as a transition system, where program execution is described in particular steps using transition relations [12, 21, 32, 38]; thus the behavior of programs is defined. And this method is our subject of interest in this paper.

Another approach connected to small-step approach is the K framework. It is an executable rewriting-based semantic definitional framework in which programming languages, calculi, as well as type systems or formal analysis tools can be defined, making use of configurations, computations, and rules. It was introduced by Grigore Roşu in 2003 [26]. The framework consists of two components: general-purpose concurrent rewriting approach (K rewriting) and a definitional technique specialized for concurrent programming languages or systems (K technique) which yields a semantic definitional style [28]. Configurations organize the state in units called cells (K cell structures), which are labeled and can be nested. Computations carry computational meaning as special nested list structures sequentializing computational tasks, such as fragments of program. The rewriting rules of the K framework make it explicit which parts of the term they read-only, write-only, read-write, or do not care about. A complete K definition of *IMP* (*While*) has been presented in [27]. Furthermore, an extended language *IMP++* is presented with the increment construct $++x$ which introduces a side effect for expressions (as in C-like languages), then *print* construct and the *input* constructs are introduced, and halting of program (internal and external ones) is added into the definition of *IMP++*.

A further particular type of small-step semantics is Reduction Semantics with Evaluation Contexts (RSEC), also known as contextual semantics, which models execution as a sequence of atomic rewrites of state, between each of which some small amount of time passes. Derivations are expressed as sequences that progress with time, rather than as trees of inference that conclude instantaneously. Reduction semantics was introduced by Matthias Felleisen and colleagues in 1992 [5]. The evaluation context style improves over small-step structural operational semantics in two ways: it gives a more compact semantics to context-sensitive reduction, by using parsing to find the next redex (a term that can be transformed in a single step); and it provides the possibility to also modify the context in which a reduction occurs, making it much easier to deal with control-intensive features. Here, an evaluation context is a term with a "hole" (a placeholder) in the place of a subterm. Additionally, one can also include the configuration as a part of the evaluation context, and thus to have full access to semantic information "by need". The contexts allow the designer of a reduction semantics to factor the definition of calculus into one part that specifies the atomic steps of computation and the second part that controls where these steps may occur. Reduction semantics with evaluation contexts does precisely that it allows to formally define evaluation contexts; rules become mostly unconditional and reductions can only happen "in context". Reduction semantics is considered as yet another way to define single-step semantic relation [14], a small-step semantics where the atomic execution step is a rewrite of the program. For modeling programming languages, Felleisen-Hieb-style reduction semantics, and their type systems, a powerful software tool PLT Redex has been designed [6]. PLT Redex is a lightweight, embedded domain-specific programming language and it comes with a suite of tools for working with the semantics. In principle, Redex is hosted in PLT Scheme.

There are some other methods for defining formal semantics, e.g. natural semantics known also as a "big-step" operational semantics [13, 16], axiomatic semantics [9] for verification purposes with various extensions [17], action semantics [20] as a hybrid between denotational and operational semantics and game semantics [10, 11] defining a semantics by game trees and strategies; however, we do not consider these approaches in this paper.

The denotational semantics, also called mathematical semantics that expresses the meaning of a program in terms of mathematical structures and mappings between them, belongs to popular semantic methods. In a simplified case, sets are used as the semantic domains and the execution of statements is described by functions [30]. The general definition of denotational semantics uses lattices and homomorphisms between them [36]; an alternative formulation is based on relations [33]. Denotational semantics provides the results of program execution, but it does not consider the details during the execution process.

In the last decades, categories have become useful mathematical structures for modeling programs and program systems [2, 18, 22]. Categories enable to work not only with sets that are carriers of the most mathematical structures but also with more complex structures that are often used in computer science [23]. The morphisms can express the changes between these structures. Functors, the morphisms between categories, express mappings between categorical structures; they are suitable for describing useful properties of systems. In [34], we have defined a categorical denotational semantics of a procedural language, where we constructed a category of states as a model of a language.

In this paper, we construct a coalgebraic semantics of an imperative language. It is a further contribution to our research project to prepare a package of modules serving to define the semantics of programs by several semantic methods. This paper together with our previous work in [34] presents a theoretical foundation for this package and is designed to be easily implementable. Therefore we try to use similar notions that can be implemented uniformly. But the coalgebraic approach requires to define behavior of programs step by step and it induces fundamental changes and necessity of new concepts different from the approach published in [34].

Categories of states are the bases for important and useful mathematical structures: algebras and coalgebras. The objects of these categories form a state space and their morphisms express the changes of states. Polynomial endofunctors over these categories characterize different kinds of systems and they model the changes of states. If we assume a state space X and a polynomial endofunctor Q over a state space, then an algebra a is defined as a mapping $a : Q(X) \to X$ and a coalgebra c, as its dual notion, is defined as $c : X \to Q(X)$. While algebras are useful for modeling the construction of programs, coalgebras enable to model their behavior. Our own results on the coalgebraic approach were published in [19, 35]. There are several publications defining coalgebraic semantics. The main ideas about coalgebraic approach come from [29]. The author considers coalgebras as transition systems describing the execution of programs in particular steps. Some additionally used language elements mentioned above are elaborated in a few works, e.g. declarations in [7, 31] and input/output using process algebras in [3, 8]. Also in the categorical approach, many publications define coalgebras for simplified kinds of systems and they ignore some details occurring in real ones.

We present a new approach on how to define the coalgebraic semantics of an imperative language containing the major features of a real imperative programming language with common statements, variable declarations, and input/output statements. We construct our coalgebra gradually from the signatures, their representations, base category, polynomial endofunctor up to coalgebraic representation. This detailed definition is more understandable also for practical programmers. The graphical representation of coalgebra can provide a good background also for educational purposes of young IT experts, and it seems to be easily implementable within our package of semantics.

In the next section, we present a short overview of the traditional definition of an operational semantics for a simple imperative language well-known as language *While* or *IMP*, presented e.g. in [21]. We adopted the structure of this language, and for pedagogical reasons, we refer to this language as *Jane.* In Section 2, we introduce traditional definition of operational semantics. Basic concepts and definitions for coalgebras are in Section 3. Then we extend our language to *E-Jane* (Section 4) with the additional constructs that are common in imperative languages. Our extended language *E-Jane* does not contain only the five basic statements of Dijkstra's language (variable assignment statement, empty statement, composed statement, conditional statement, and loop statement) but also variable declarations, input and output statements, and a block statement. To define the operational semantics in coalgebraic terms, we define the concepts of memory abstraction, statement list, declaration list, and configuration as abstract data types (Section 5). We represent these abstract types so that a representation of a configuration contains the information what is to be executed together with an actual snapshot of a memory, and lists of input and output. We consider the set of configurations as our state space. In Section 6, we define how statements of a program are executed; in Section 7, we give the semantics of declarations as the morphisms between configurations. In Section 8, we construct a category of configurations, suitable polynomial endofunctor Q and Q-coalgebra. We finish the paper with a simple example illustrating our approach (Section 9).

2 TRADITIONAL DEFINITION OF OPERATIONAL SEMANTICS

Before we explain our approach of defining a coalgebraic operational semantics, an overview of the traditional definition of operational semantics is given. We introduce a simple imperative language *Jane* consisting of expressions (arithmetic and Boolean ones) and statements; then we give it an operational semantics.

2.1 The Language Jane

The formal syntax of *Jane* has been inspired by the formal syntax of *While* [21]. The language *Jane* is considered as a folklore (toy) language, without an official inventor; it has been used in many textbooks and papers, often with slight syntactic variations. The syntax of the language is described by syntactic domains and production rules. We consider the following syntactic domains:

- $n \in$ **Num** – numerals (digit strings);
- $x \in$ **Var** – variable names;
- $e \in$ **Aexpr** – arithmetic expressions;
- $b \in$ **Bexpr** – Boolean expressions;
- $S \in$ **Statm** – statements.

The elements of **Num** and **Var** have no internal structure significant for the semantics. The syntactic domain **Aexpr** consists of all well-formed arithmetic expressions created by the following syntax:

$$e ::= n \mid x \mid e + e \mid e - e \mid e * e. \tag{1}$$

A Boolean expression from **Bexpr** can be of the following structure:

$$b ::= \texttt{false} \mid \texttt{true} \mid e = e \mid e \leq e \mid \neg b \mid b \wedge b. \tag{2}$$

As elements S of the syntactic domain **Statm**, we consider Dijkstra's five elementary statements, namely variable assignment, empty statement, sequential composition of statements, conditional statement, and loop statement:

$$S ::= x := e \mid \texttt{skip} \mid S; S \mid \texttt{if } b \texttt{ then } S \texttt{ else } S \mid \texttt{ while } b \texttt{ do } S. \tag{3}$$

2.2 Operational Semantics of Jane

A structural operational semantics is defined as a transition system that describes each step of a program execution using transition relations. The traditional approach to this method defines transition relations by inference rules that describe the changes in memory during a program execution. As some abstraction of computer memory, the concept of a "state" is used. The set **State** of states is the basic semantic domain and its elements are functions from variables to values. For simplicity, we consider that all variables are implicitly typed as integer values from the set **Z**. Thus a state s is defined as a function

$$s : \mathbf{Var} \to \mathbf{Z}. \tag{4}$$

A change of a state means an actualization of a state which is written using a substitution. If a value of a variable y is changed to some new value $\mathbf{n}$, then a new state s' is defined as

$$s' = s[y \mapsto \mathbf{n}]. \tag{5}$$

This means that a new state s' is the same as s excluding the value of y, which was changed (substituted) to a new value $\mathbf{n} \in \mathbf{Z}$. Formally:

$$s'x = (s[y \mapsto \mathbf{n}])x = \begin{cases} \mathbf{n}, & \text{if } x = y; \\ sx, & \text{if } x \neq y. \end{cases} \tag{6}$$

$\llbracket e \rrbracket : \mathbf{State} \to \mathbf{Z}$	$\llbracket b \rrbracket : \mathbf{State} \to \mathbf{Bool}$
$\llbracket n \rrbracket s = \mathbf{n}$	$\llbracket \mathtt{true} \rrbracket s = \mathbf{true}$
$\llbracket x \rrbracket s = s\ x$	$\llbracket \mathtt{false} \rrbracket s = \mathbf{false}$
$\llbracket e_1 + e_2 \rrbracket s = \llbracket e_1 \rrbracket s \oplus \llbracket e_2 \rrbracket s$	$\llbracket e_1 = e_2 \rrbracket s = \begin{cases} \mathbf{true}, & \text{if } \llbracket e_1 \rrbracket s {=} \llbracket e_2 \rrbracket s \\ \mathbf{false}, & \text{otherwise} \end{cases}$
$\llbracket e_1 - e_2 \rrbracket s = \llbracket e_1 \rrbracket s \ominus \llbracket e_2 \rrbracket s$	$\llbracket e_1 \leq e_2 \rrbracket s = \begin{cases} \mathbf{true}, & \text{if } \llbracket e_1 \rrbracket s {\leq} \llbracket e_2 \rrbracket s \\ \mathbf{false}, & \text{otherwise} \end{cases}$
$\llbracket e_1 * e_2 \rrbracket s = \llbracket e_1 \rrbracket s \otimes \llbracket e_2 \rrbracket s$	$\llbracket \neg b \rrbracket s = \begin{cases} \mathbf{true}, & \text{if } \llbracket b \rrbracket s {=} \mathbf{false} \\ \mathbf{false}, & \text{otherwise} \end{cases}$
	$\llbracket b_1 \wedge b_2 \rrbracket s = \begin{cases} \mathbf{true}, & \text{if } \llbracket b_1 \rrbracket s {=} \llbracket b_2 \rrbracket s {=} \mathbf{true} \\ \mathbf{false}, & \text{otherwise} \end{cases}$

Table 1. Semantics of arithmetic and Boolean expressions

Arithmetic and Boolean expressions serve for computing values of two implicit types of the language *Jane*, the type of integer values and the type of Boolean values, respectively. In defining the semantics of both types of expressions, an actual state is used but not changed in the process of evaluation. So the state plays only a passive rôle in the evaluation of expressions. The semantic domain for arithmetic expressions is the set $\mathbf{Z}$ of integer numbers; for Boolean values, we introduce a new semantic domain **Bool** containing two elements – **true** and **false**:

$$\mathbf{Bool} = \{\mathbf{true}, \mathbf{false}\}. \tag{7}$$

Table 1 defines the semantic functions $[\![e]\!]$ and $[\![b]\!]$ mapping arithmetic expressions respectively Boolean expressions to functions from states to integer values respectively Boolean values. These functions produce transient data that are consumed during the program execution and whose values are never directly stored into memory except by assigning them to variables.

The changes of states are defined for particular statements by inference rules. An inference rule consists of a finite number of assumptions and a conclusion:

$$\frac{assumption_1, \ldots, assumption_n}{conclusion}\ (rule\_name)$$

$$\langle x := e, s\rangle \Rightarrow s[x \mapsto [\![e]\!]s]\ (1_{os}) \qquad \langle \mathtt{skip}, s\rangle \Rightarrow s\ (2_{os})$$

$$\frac{\langle S_1, s\rangle \Rightarrow \langle S_1', s'\rangle}{\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s'\rangle}\ (3^1_{os}) \qquad \frac{\langle S_1, s\rangle \Rightarrow s'}{\langle S_1; S_2, s\rangle \Rightarrow \langle S_2, s'\rangle}\ (3^2_{os})$$

$$\frac{[\![b]\!]s = \mathbf{true}}{\langle \mathtt{if}\ b\ \mathtt{then}\ S_1\ \mathtt{else}\ S_2, s\rangle \Rightarrow \langle S_1, s\rangle}\ (4^{\mathbf{true}}_{os})$$

$$\frac{[\![b]\!]s = \mathbf{false}}{\langle \mathtt{if}\ b\ \mathtt{then}\ S_1\ \mathtt{else}\ S_2, s\rangle \Rightarrow \langle S_2, s\rangle}\ (4^{\mathbf{false}}_{os})$$

$$\langle \mathtt{while}\ b\ \mathtt{do}\ S, s\rangle \Rightarrow \langle \mathtt{if}\ b\ \mathtt{then}\ (S; \mathtt{while}\ b\ \mathtt{do}\ S)\ \mathtt{else}\ \mathtt{skip}, s\rangle\ (5_{os})$$

Table 2. Semantics of statements

The assumptions and the conclusion are transition relations between particular configurations. A configuration

$$\alpha = \langle S, s\rangle \tag{8}$$

expresses that a statement S is to be executed in a state s. A transition has form

$$\alpha \Rightarrow \alpha'. \tag{9}$$

Here α' can be either a state, if the statement is executed in one step, or a configuration $\langle S', s'\rangle$, where S' stands for a statement that is to be executed in the following step(s). So a transition describes an one-step action [37].

The inference rules for *Jane* in structural operational semantics [21] are given in Table 2. An operational semantics can also be defined for block statements with declarations of local variables, e.g. in [24]. Our short overview of operational semantics serves only for illustration of the traditional approach, and we will treat these language constructions fully in our coalgebraic approach.

3 BASIC NOTIONS OF COALGEBRAS

Coalgebras are a useful tool for modeling the behavior of dynamic systems. They are defined over a base category whose objects create a state space and whose category morphisms are transitions.

To define a coalgebra for a system, we start with the signature of an abstract data type of states. A signature contains operation symbols defined on data types that can be:

- constructors – these generate the algebraic data types; they "work into" the data types;
- selectors (destructors) – these describe changes of states; they are also called behavioral operations, because they can provide observable values; they "work out" of a data structure;
- derived operations – these help to work with corresponding data structures.

Selectors play the most important rôle among the operation symbols for constructing coalgebras. They are interpreted as morphisms in a base category of states.

The dynamics, i.e. the execution of particular steps, is supplied by a polynomial endofunctor

$$F : \mathscr{C} \rightarrow \mathscr{C} \tag{10}$$

defined over a category $\mathscr{C}$ of states; it is indicated by the corresponding signature. The notion of a polynomial endofunctor [15] comes from its shape that is similar to that of polynomials, because it can be constructed from constants, products, coproducts and exponentials, e.g.:

$$FX = A_0 + A_1 \times X^{B_1} + A_2 \times X^{B_2} + \ldots + A_n \times X^{B_n}. \tag{11}$$

Here, X stands for a state space, the A_i, for $i = 0, \ldots, n$, are sets of observable values, the B_i are some fixed sets and $\times$ and $+$ are the operations on category objects: products and sums, respectively. The concrete shape of a polynomial endofunctor determines (characterizes) a particular kind of systems.

Then a coalgebra can be defined as a mapping from the state space to the result of the endofunctor applied to this state space; this mapping can be represented as a tuple of selectors:

$$\langle [\![sel_1]\!], \ldots, [\![sel_n]\!] \rangle : X \rightarrow FX. \tag{12}$$

Based on these general definitions, we show in the following sections how a coalgebra defining the operational semantics of an imperative programming language can be constructed.

4 EXTENDED IMPERATIVE LANGUAGE *E-JANE*

Because our aim is to treat a language with the constructs available in real programming languages, we extend the language introduced in Section 2 to the language *E-Jane* that contains the constructs common in the most imperative programming languages.

Assume a set **Decl** as a syntactic domain of declarations. Each variable used in a program has to be declared. We introduce a declaration $D \in \mathbf{Decl}$ as

$$D ::= \texttt{var}\ x. \tag{13}$$

We assume that the variables are implicitly of the integer type. This restriction enables us to focus on the main ideas of our approach.

We add three further statements: a block statement enclosed in the brackets `begin` and `end`; an input statement `read`, and an output statement `print`:

$$S ::= \dots \mid \texttt{begin}\ D_1; \dots; D_k; S\ \texttt{end} \mid \texttt{read}\ x \mid \texttt{print}\ e. \tag{14}$$

where $D_1; \dots; D_k$ is a finite list of local declarations and $k \in \mathbf{N}_0$. The local declarations are visible only inside a given block. We have to take into consideration also the program global declarations that are located at the beginning of a program and they are visible within a whole program.

A program in *E-Jane* has then a form for $k \in \mathbf{N}_0, n \in \mathbf{N}$:

$$D_1; \dots; D_k; S_1; \dots; S_n, \tag{15}$$

i.e., it is a finite list of global declarations followed by a finite list of statements. So *E-Jane* becomes closer to many real imperative languages.

The following sections describe how we construct an operational semantics by a coalgebra for this language.

5 MEMORY AND ITS REPRESENTATION

Observing the behavior of a program during its execution means to define how a program is being executed step by step and how the snapshots of the memory are being changed in detail. First, we specify a structure (a data type) for the state space. We consider as our state space the data type *Config* of configurations for which we introduce a signature.

Each configuration is a tuple whose first item is a finite list of declarations together with a list of statements to be executed; its second item is the actual memory content. The third and the fourth items are lists of input and output values, respectively. We start with the signature for the lists of declarations and statements; we follow with the signature for memory; then we construct a signature of configurations. We put emphasis on the selector operations that play an important rôle in coalgebras. Then we define a suitable representation of the specified types.

5.1 Signature of Configurations

As we have mentioned, a program in the imperative language *E-Jane* is a finite list of declarations of global variables followed by a finite list of statements. Declarations do not affect the computer memory content, but they are important because they reserve cells for declared variables.

We start with specifying a parametrized signature [4] of finite lists, then we define its two instantiations for the declaration list and statement list. Then we specify a signature for abstract memory. Using these three signatures, we specify the data type *Config* of configurations.

$$\begin{array}{l} \Sigma_{List} = List_{fin}\,[Item] \\ types : List, Item \\ opns : \\ \quad init : \ \to Item \\ \quad head : List \to Item \\ \quad tail : List \to List \end{array} \tag{16}$$

The operation *init* creates the empty list of items. The operation *head* extracts the first item and *tail* returns the rest of a list of items.

Let *Decl* and *Statm* be the type names for declarations and statements, respectively. Then we define the signature of declaration lists by setting *Item* = *Decl* as the instantiation

$$\Sigma_{Decl\_List} = \Sigma_{List}\,[Decl]\,.$$

Similarly, setting *Item* = *Statm* we get the signature of statement lists

$$\Sigma_{Statm\_List} = \Sigma_{List}\,[Statm]\,.$$

An abstract memory is a basic concept in the semantics of imperative languages. It is often called *state* in many publications, but for our purposes the notion *memory* is more appropriate. Each variable occurring in a program has to be allocated, i.e., a memory cell is reserved and named by the elaboration of a declaration. The value of an allocated variable can be assigned and modified inducing a change of the actual memory.

According to these ideas about the concept of abstract memory, we formulate a signature Σ_{Memory} as an abstract data type which uses types *Var* and *Value* for variables and values, respectively. This signature consists of types and operation specifications on the type *Memory*:

$$\begin{array}{lll} \Sigma_{Memory} = & & \\ & types: & Memory, Var, Value \\ & opns: & init: \;\to Memory \\ & & alloc: Var, Memory \to Memory \\ & & get: Var, Memory \to Value \\ & & del: Memory \to Memory \end{array} \tag{17}$$

The operation specifications have the following intuitive meaning (the following subsection will explain the notion of "nesting levels" in more detail):

- *init* merely creates the initial memory of a program;
- *alloc* reserves a new memory cell for a variable at its actual nesting level;
- *get* returns a variable value in a given memory cell at its actual nesting level;
- *del* deallocates (releases) all memory cells for the variables declared on the highest nesting level.

The language *E-Jane* contains the statements for user input and output. Input and output values are of the type *Value* and they form the lists. The corresponding signatures are

$$\Sigma_{Input} = \Sigma_{List}\,[I\_Value]\,, \tag{18}$$

$$\Sigma_{Output} = \Sigma_{List}\,[O\_Value] \tag{19}$$

where *I_Value* and *O_Value* are type names for input and output values, respectively.

To specify the process of program execution we introduce a new type *Config* by its signature:

$$\begin{array}{lll} \Sigma_{Config} = & \multicolumn{2}{l}{\Sigma_{Decl\_List} + \Sigma_{Statm\_List} + \Sigma_{Memory} + \Sigma_{Input} + \Sigma_{Output} +} \\ & types: Config & \\ & opns: & next: Config \to Config \\ & & read: Config, I\_Value \to Config \\ & & print: Config \to O\_Value, Config \end{array} \tag{20}$$

This data type introduces a new type *Config* with its operations (transition functions), where

- *next* provides the next configuration;
- *read* takes an input value and stores it in a corresponding memory cell;
- *print* computes a value of an argument and produces it as an observable value.

5.2 Representation of Types

Now we assign a representation to the data types specified above. First, we represent the unstructured data type *Value* as a set of integers $\mathbf{Z}$ together with the undefined value $\perp$:

$$\mathbf{Value} = \mathbf{Z} \cup \{\perp\}\,. \tag{21}$$

We assign to the type *Var* a countable set **Var** of variable names. To deal with nested block statements, we extend this set with special (dummy) variables `begin` and `end` that are not declared. An undefined variable $\perp$ is also in **Var**, but it serves only for the initial memory of a program.

We represent the type *List* in Σ_{Decl_List} as a set of finite lists **Decl_List**, which elements are finite lists of declarations D^*. Analogously, the type *State* in Σ_{Statm_List} is represented also as a set of finite lists **Statm_List** with elements S^* standing for the finite lists of statements. That means

$$D^* = D_1; \ldots ; D_m, \tag{22}$$

$$S^* = S_1; \ldots ; S_n, \tag{23}$$

where $D_i \in \mathbf{Decl}$, for $i = 1, \ldots, m$, and $S_i \in \mathbf{Statm}$, for $i = 1, \ldots, n$. The representations of the operation symbols *head* and *tail* are defined as it is regular for lists.

Because our language contains also a block statement, possibly with local variables declarations, we will consider the nesting level of a block. This nesting level allows us to create a variable environment, the notion known from operational semantics, and it enables us to distinguish local declarations from global ones. Therefore we introduce the set **Level** of nesting levels denoted by natural numbers l:

$$l \in \mathbf{Level}, \quad \text{where} \quad \mathbf{Level} = \mathbf{N}. \tag{24}$$

We assign to the type *Memory* the set **Memory** of all possible non-empty memory contents:

$$\mathbf{Memory} = \{m : \mathbf{Var} \times \mathbf{Level} \to \mathbf{Value}\}\,.$$

Each memory m expresses one moment of program execution, an actual snapshot of a computer memory. The function m is identified with its graph [30], $graph(m)$, i.e., a set of pairs, where the first member of each pair is an argument of this function and the second member is the value of the function:

$$graph(m) = \{((x,l),v) \mid (x,l) \in dom(m) \wedge m(x,l) = v\}\,. \tag{25}$$

For a visualization of an actual memory, we can write the function m as a table with possibly unfilled cells denoted by $\perp$ expressing an undefined value for a declared variable. This visualization increases the readability; it is illustrated on the left-hand side in Figure 1.

variable	level	value
x_1	1	v_1
$\vdots$		
x_n	l	v_n

variable	level	value
$\perp$	1	$\perp$

Figure 1. Visualization: actual memory and initial memory

Now we define the representations of the operation specifications from the signature Σ_{Memory} as follows. The operation specification *init* is represented by a function $[\![init]\!]$ defined by

$$[\![init]\!] = m_0 = \{((\perp, 1), \perp)\} \tag{26}$$

which creates an initial memory m_0 of a program, with no declared variable. Its rôle is only to set the nesting level to a value 1 at the beginning of program execution as it is on the right-hand side in Figure 1.

For defining the representation of the further operations on a memory, we need to specify an actual (maximal) level of a nesting. Let

$$m = \{((x_1, l_1), v_1), \dots ((x_m, l_n), v_k)\}$$

be an actual memory. We define an auxiliary function

$$maxlevel : \mathbf{Memory} \to \mathbf{Level} \tag{27}$$

defined by

$$maxlevel(m) = L, \tag{28}$$

such that $\exists j = 1, \dots n. L \geq l_j$.

The operation $[\![alloc]\!]$ adds a new item to the memory m (creates a new entry in the table); it is defined as

$$[\![alloc]\!](x, m) = m \cup \{((x, maxlevel(m)), \perp)\}. \tag{29}$$

This operation sets the actual nesting level l to the declared variable (left table in Figure 2).

The operation $[\![get]\!]$ returns the value of a variable declared on the highest nesting level. We introduce an abbreviation *Highest* which expresses the maximum nesting level where a variable x is declared:

$$Highest(m, x) = \max \{l' \mid l' \in \mathbf{Level} \text{ and } (x, l') \in dom(m)\};$$

For simplification of the formulation, we define the following predicate:

$$Defined(m, x) \equiv_{def} \exists l' \in \mathbf{Level}.(x, l') \in dom(m)$$

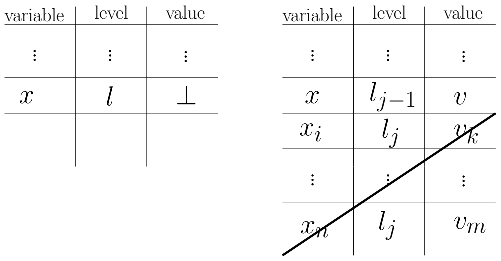


Figure 2. Variable allocation and deallocation

that expresses whether the variable x is declared in an actual memory m.

The operation $[\![get]\!]$ is then defined as

$$[\![get]\!](x,m) = \begin{cases} m(x,l), & \text{where } l = \mathit{Highest}(m,x), \\ & \text{if } \mathit{Defined}(m,x); \\ \perp, & \text{otherwise.} \end{cases} \tag{30}$$

The operation $[\![del]\!]$ deallocates (releases from the table) all the variables declared on the highest nesting level (right table in Figure 2):

$$[\![del]\!]m = m \setminus \{((x, \mathit{maxlevel}(m)), v) \mid x \in \mathbf{Var} \wedge v \in \mathbf{Value}\}. \tag{31}$$

We also consider a special memory content

$$m_\perp = ((\perp, 0), \perp) \tag{32}$$

expressing the undefined memory content when a program aborts.

The representation of the type *List* in Σ_{Input} is a finite list $i^* = i_1; \ldots; i_m$. Similarly, the type *List* in Σ_{Output} is a finite list $o^* = o_1; \ldots; o_n$, where $i_j, o_k \in \mathbf{Value}$. For simplicity, we consider here only finite lists of input and output values. These lists form the semantic domains **Input** of lists of input values and **Output** of lists of output values.

The last type to be represented is *Config*. We represent it by a set **Config** of configurations as a cartesian product

$$\mathbf{Config} = \mathbf{Program} \times \mathbf{Memory} \times \mathbf{Input} \times \mathbf{Output}, \tag{33}$$

where **Program** consists of the lists of global declarations/statements to be yet elaborated and/or executed. A configuration is a quadruple

$$\mathit{config} = ([\![D^*; S^*]\!], m, i^*, o^*) \tag{34}$$

for $m \in \mathbf{Memory}$, $i^* \in \mathbf{Input}$ and $o^* \in \mathbf{Output}$. We consider the representation **Config** as the state space; its elements will be the objects in the base category of our coalgebra. We note that we consider at the beginning of program execution that o^* is empty, $o^* = \varepsilon$.

The most important operations for defining the operational semantics by a coalgebra are the transition operations $[\![next]\!]$, $[\![read]\!]$ and $[\![print]\!]$ that we will define later. These are the morphisms in our category of configurations.

In the next section, we define how a step of the execution of statements can be defined as a morphism between configurations.

6 EXECUTION OF STATEMENTS

The interpretation of the operations *next*, *read*, and *print* represents the execution of statements. Here we discuss how each of these operations can be defined. We note that in operational semantics we model only one step of a statement execution.

Generally, the interpretation of *next* is a mapping

$$[\![next]\!] : \mathbf{Config} \rightarrow \mathbf{Config}. \tag{35}$$

The finite list S^* of statements is changed in each execution step according to the correspondingly executed statement. The first member of this sequence is a statement to be executed in a given memory m. Now we define the function $[\![next]\!]$ for each kind of statement.

The assignment statement $x := e$ is to be executed in a memory m in one step. The operation $[\![next]\!]$ for this statement is defined as

$$[\![next]\!]([\![x := e; S^*]\!], m, i^*, o^*) = ([\![S^*]\!], m', i^*, o^*) \tag{36}$$

where

$$m' = \begin{cases} m\,[((x, \mathit{Highest}(m,x)), v) & \mapsto \quad ((x, \mathit{Highest}(m,x)), [\![e]\!]m)]\,, \\ & \quad\ \text{if } \mathit{Defined}(m,x); \\ m_{\perp}, & \quad\ \text{otherwise.} \end{cases}$$

The transition mapping returns the tail of the statement list together with a new memory that contains a new value $[\![e]\!]m$ for the variable x.

In like manner, we define the semantics for the empty statement `skip` that executes also in one step but without any change of memory. Its semantics is defined as a morphism without change of a memory m:

$$[\![next]\!]([\![\texttt{skip}; S^*]\!], m, i^*, o^*) = ([\![S^*]\!], m, i^*, o^*). \tag{37}$$

This statement is considered as an identity on memory. On the other hand, this morphism is not an identity on configuration because of shortening of the statement list.

To define the semantics of a sequence of statements, we need to distinguish two situations. Assume that the rest of the program has a form

$$S_1; S_2; S^*. \tag{38}$$

The role of operational semantics is to define only the first execution step. A statement S_1 can be executed either in one step or in more steps. In the first case, e.g., if it is a statement being executed in one step, after this step the statements $S_2; S^*$ remain to be executed. In the second case, after the first step a sequence $S_1'; S_2; S^*$ has to be executed. Thus we define the mapping $[\![next]\!]$ for these situations as

$$[\![next]\!]([\![(S_1; S_2); S^*]\!], m, i^*, o^*) = \begin{cases} ([\![S_2; S^*]\!], m', i'^*, o'^*), \\ \qquad \text{if } \langle S_1, m\rangle \Rightarrow m'; \\ ([\![S_1'; S_2; S^*]\!], m', i'^*, o'^*), \\ \qquad \text{if } \langle S_1, m\rangle \Rightarrow \langle S_1', m'\rangle. \end{cases} \tag{39}$$

The memory m' depends on the actual execution of the statement S_1. The lists i'^* and o'^* represent the situation when the statement S_1 stands for a user input or output.

Consider now that the first statement to be executed is a conditional statement S as the first statement in the list $S; S^*$:

$$S = \texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2. \tag{40}$$

The first step depends on the value of the Boolean expression b in the actual memory m. If $[\![b]\!]m = \mathbf{true}$, then the execution follows with the statement S_1; and with the statement S_2, otherwise. We note that the execution of the conditional statement is still deterministic and the memory m is not changed during this first step. Therefore, the semantics of conditional is

$$\begin{gathered} [\![next]\!]([\![\texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2; S^*]\!], m, i^*, o^*) = \\ \begin{cases} ([\![S_1; S^*]\!], m, i^*, o^*), & \text{if } [\![b]\!]m = \mathbf{true}; \\ ([\![S_2; S^*]\!], m, i^*, o^*), & \text{if } [\![b]\!]m = \mathbf{false}. \end{cases} \end{gathered} \tag{41}$$

The first step of the execution of the loop statement `while` b `do` S is the same as the execution of the following conditional statement:

$$\begin{aligned} &[\![next]\!]\left([\![\texttt{while } b \texttt{ do } S; S^*]\!], m, i^*, o^*\right) \\ &\quad = \left([\![\texttt{if } b \texttt{ then } S; \texttt{while } b \texttt{ do } S \texttt{ else skip}; S^*]\!], m, i^*, o^*\right). \end{aligned} \tag{42}$$

Now we define the semantics of the user input statement `read` x. It is executed in one step and it assigns to a declared variable x an input value v, i.e., it changes a configuration as described by the transition function

$$[\![read]\!] : \mathbf{Config} \to \mathbf{Config}^{\mathbf{Value}} \tag{43}$$

defined as

$$[\![read]\!]([\![\texttt{read}\ x; S^*]\!], m, i^*, o^*) = \begin{cases} \lambda v'.([\![S^*]\!], m', \mathit{tail}(i^*), o^*), \\ \quad \text{if } \mathit{Defined}(m, x); \\ ([\![S^*]\!], m_\perp, \mathit{tail}(i^*), o^*), \\ \quad \text{otherwise}, \end{cases} \tag{44}$$

where $m' = m[((x, \mathit{Highest}(m,x)), v) \mapsto ((x, \mathit{Highest}(m,x)), v')]$. That means that any value stored in a variable x declared on the highest nesting level is changed to the input value v'.

The output statement `print` e is also executed in one step. It computes a value of an arithmetic expression e in an actual memory m and it provides a result as an observable value. A memory is not changed but the configuration is. The semantics of this statement can be described by the transition function

$$[\![print]\!] : \mathbf{Config} \to \mathbf{Value} \times \mathbf{Config} \tag{45}$$

defined as

$$[\![print]\!]([\![\texttt{print}\ e; S^*]\!], m, i^*, o^*) = ([\![e]\!]m, ([\![S^*]\!], m, i^*, [\![e]\!]m; o^*)) \,. \tag{46}$$

The first step of the execution of a block statement $S = \texttt{begin}\ D^*; S'\ \texttt{end}$ is modeled as

$$\begin{gathered} [\![next]\!]([\![\texttt{begin}\ D^*; S'\texttt{end}; S^*]\!], m, i^*, o^*) = \\ ([\![D^*; S'\ \texttt{end}; S^*]\!], [\![begin]\!]m, i^*, o^*) \end{gathered} \tag{47}$$

where the special declaration `begin` (see in Section 7) is elaborated and the execution proceeds with elaborating the local declarations and executing the body of the block.

For indicating that an execution step yields an undefined result, we introduce an auxiliary mapping

$$[\![abort]\!] : \mathbf{Config} \to \mathbf{Config} \tag{48}$$

defined as

$$[\![abort]\!](\mathit{config}) = (\varepsilon, m_\perp, \varepsilon, \varepsilon) \tag{49}$$

where ε denotes the empty list. This definition ensures that an aborted program stops in a stuck configuration that does not contain any statements to be executed.

In the next section, we define the elaboration of declarations in a uniform way.

7 SEMANTICS OF DECLARATIONS

Each variable occurring in an *E-Jane* program has to be declared. Declarations are elaborated, i.e., a new memory cell is allocated on the actual nesting level with the undefined value. Declarations form a finite list $D_1; D_2, \dots ; D_n$, where each $D_i, i = 1, \dots , n$ has the structure $\texttt{var}\ x_i$.

We represent each declaration as a function on a memory:

$$[\![\texttt{var}\ x]\!] : \mathbf{Memory} \to \mathbf{Memory}. \tag{50}$$

We define it for a given memory m as follows:

$$[\![\texttt{var}\ x]\!]m = [\![alloc]\!](x, m). \tag{51}$$

This definition expresses that a declaration is elaborated in one step.

Declarations may appear in a program either on the global level with $l = 1$ or in a block statement with a nesting level $l > 1$. In case of global declarations, new entries are allocated with the initial nesting level $l = 1$; for instance, for a variable x on the level $l = 1$ the following entry is being created:

$$((x, 1), \bot). \tag{52}$$

To supply the incrementation of a nesting level in the case of a block statement, we introduce two special variables: $\texttt{begin}, \texttt{end} \in \mathbf{Var}$. A fictive declaration $\texttt{begin}$ serves to bound the locally declared variables to an incremented nesting level while $\texttt{end}$ indicates the end of a block statement where locally declared variables are released from the table. These declarations are also elaborated in one step by functions $[\![begin]\!]$ and $[\![end]\!]$ on a memory m:

$$[\![begin]\!], [\![end]\!] : \mathbf{Memory} \to \mathbf{Memory}, \tag{53}$$

defined as follows:

$$\begin{aligned} [\![begin]\!]m &= m \cup \{((\texttt{begin}, maxlevel(m) + 1), \bot)\}, \\ [\![end]\!]m &= [\![del]\!]m. \end{aligned} \tag{54}$$

Because the objects of our categorical model are configurations and the elaboration of a declaration affects a given configuration, we define a morphism $[\![next]\!]$ for declarations as

$$[\![next]\!] : \mathbf{Config} \rightharpoonup \mathbf{Config}, \tag{55}$$

where

$$\begin{aligned} [\![next]\!]([\![\texttt{var}\ x; D^*; S^*]\!], m, i^*, o^*) &= ([\![D^*; S^*]\!], [\![\texttt{var}\ x]\!]m, i^*, o^*), \\ [\![next]\!]([\![\texttt{begin}\ D^*; S'\ \texttt{end}; S^*]\!], m, i^*, o^*) &= ([\![D^*; S'\ \texttt{end}; S^*]\!], [\![begin]\!]m, i^*, o^*), \\ [\![next]\!]([\![\texttt{end}; S^*]\!], m, i^*, o^*) &= ([\![S^*]\!], [\![end]\!]m, i^*, o^*). \end{aligned} \tag{56}$$

However, a morphism $[\![next]\!]$ is always defined for declarations, it can be undefined only for statements, as we showed in Section 6. So the semantics of declarations corresponds with the meaning of declarations in traditional operational semantics, i.e., each declaration actualizes an environment of variables.

8 COALGEBRA FOR THE LANGUAGE *E-JANE*

In the previous sections, we defined the notions necessary for the construction of a base category for a coalgebra. Now we construct the category $\mathscr{C}onfig$ consisting of

- configurations $config = ([\![D^*; S^*]\!], m, i^*, o^*)$ as the category objects;
- mappings $[\![next]\!], [\![read]\!], [\![print]\!]$ and $[\![abort]\!]$ as the category morphisms.

The objects in this category form the state space for our coalgebra and the morphisms are transition mappings, each of them modeling one step of program execution.

We check whether so defined structure is a category:

- Each object has to have an identity morphism. However, no morphism defining the operational semantics of *E-Jane* is an identity. To satisfy this category property, we need to define explicitly that each object has an identity morphism id_{config}.
- A composition of two composable morphisms is a morphism in $\mathscr{C}onfig$, e.g., for the execution of a sequence of statements.
- A composition of morphisms is associative, trivially.

Our category $\mathscr{C}onfig$ has a terminal object, the undefined configuration

$$config_{\perp} = (\varepsilon, m_{\perp}, \varepsilon, \varepsilon) \tag{57}$$

that indicates aborting of a program. From any object in $\mathscr{C}onfig$ there exists a unique morphism to this object because the running program can abort in any step; so $config_{\perp}$ is a terminal object in $\mathscr{C}onfig$. The category has no initial object because the starting of execution depends on an actual program that should be executed. Therefore the initial configuration is different for each program.

The execution of a loop statement is modeled as a path of morphisms, i.e. a composition of morphisms modeling the particular steps of an execution. When the loop is executed in a finite number of steps, we get some final configuration and the execution of a program can follow. When this path is infinite, we need to ensure that our model is a category, i.e., there exists an object that is a composition of an infinite path. Thus our category $\mathscr{C}onfig$ needs to have colimits [1] for all diagrams consisting of an infinite composition of configurations. The definition of a colimit in the category $\mathscr{C}onfig$ is explained in [34].

Now we have the base category that is a model of *E-Jane*; thus we can proceed to construct a coalgebra modeling the behavior of programs written in *E-Jane*. The objects of our category form the state space of a coalgebra and the morphisms are the transition mappings. Now, we construct the polynomial endofunctor for this kind of systems. Generally, the following polynomial endofunctor seems to be appropriate for our purposes:

$$Q(\mathbf{Config}) = 1 + \mathbf{Config} + O \times \mathbf{Config} + \mathbf{Config}^I. \tag{58}$$

Here $I \subseteq \mathbf{Value}$ denotes the domain of input values and $O \subseteq \mathbf{Value}$ denotes the domain of output values of the program execution.

The operation $+$ in the definition of the functor Q expresses distinct, mutual exclusive results of the functor. We discuss the possible results:

- $Q(\mathbf{Config}) = 1$ when a program aborts, i.e. it abnormally finishes and does not return a result. This situation arises when the morphism $[\![abort]\!]$ in $\mathscr{C}\!onfig$ is performed:
$$Q(config) = [\![abort]\!](config);$$
- if $Q(\mathbf{Config}) = \mathbf{Config}$, a new configuration is achieved by an elaboration of a declaration or an execution of a statement with no input and output. This situation occurs in the category by performing the morphism $[\![next]\!]$:
$$Q(config) = [\![next]\!](config);$$
- if $Q(\mathbf{Config}) = O \times \mathbf{Config}$, a change of configuration happens together with producing some observable output value. The morphism $[\![print]\!]$ is performed:
$$Q(config) = [\![print]\!](config);$$
- if $Q(\mathbf{Config}) = \mathbf{Config}^I$, an input value $i \in I$ is read by the execution of the statement `read`
$$Q(config) = [\![read]\!](config).$$

Now we can define the Q-coalgebra for the programming language *E-Jane* as a mapping:

$$\langle [\![abort]\!], [\![next]\!], [\![print]\!], [\![read]\!] \rangle : \mathbf{Config} \to Q(\mathbf{Config}). \tag{59}$$

This coalgebra models the execution of a program in particular steps, i.e., it provides the operational semantics of any program written in *E-Jane*. We note that our coalgebra models the behavior of programs written in any programming languages containing corresponding constructs.

9 EXAMPLE

We illustrate our approach on a simple program, which uses the most of the constructs of *E-Jane*. Assume a program P:

```
var x; var y;
input x; input y;
if x <= y then begin var z;
                   z:=x; x:=y; y:=z
                 end
            else skip;
print x
```

We introduce here some abbreviations:

$$\begin{aligned}
&D_1 = \texttt{var}\ x;\ \ D_2 = \texttt{var}\ y;\\
&S_1 = \texttt{read}\ x;\ \ S_2 = \texttt{read}\ y;\\
&S_3 = \texttt{if}\ x <= y\ \texttt{then begin var}\ z;\\
&\qquad\quad z := x; x := y; y := z\ \texttt{end else skip};\\
&S_4 = \texttt{print}\ x
\end{aligned}$$

Let the input values for x and y be $\mathbf{3}$ and $\mathbf{5}$, respectively. An input list is then $i^* = (\mathbf{3}, \mathbf{5})$ and an output list is empty, $o^* = \varepsilon$.

An initial configuration is

$$\mathit{config}_0 = ([\![D_1; D_2; S_1; S_2; S_3; S_4]\!], m_0, i^*, o^*)$$

and an initial memory m_0 contains only information about starting value of declaration nesting, $m_0 = ((\perp, 1), \perp)$ (Figure 3).

m_0		
$\perp$	1	$\perp$

Figure 3. Initial memory

First, the declarations are elaborated in separate steps:

$$\begin{aligned}
Q(\mathit{config}_0) &= [\![\mathit{next}]\!](\mathit{config}_0) = \mathit{config}_1\\
&= ([\![D_2; S_1; S_2; S_3; S_4]\!], [\![\texttt{var}\ x]\!]m_0, (\mathbf{3}, \mathbf{5}), \varepsilon),\\
Q(\mathit{config}_1) &= [\![\mathit{next}]\!](\mathit{config}_1) = \mathit{config}_2\\
&= ([\![S_1; S_2; S_3; S_4]\!], [\![\texttt{var}\ y]\!]m_1, (\mathbf{3}, \mathbf{5}), \varepsilon),
\end{aligned}$$

where $m_1 = [\![\texttt{var}\ x]\!]m_0$ and $m_2 = [\![\texttt{var}\ y]\!]m_1$ (Figure 4).

m_1		
x	1	$\perp$

m_2		
x	1	$\perp$
y	1	$\perp$

Figure 4. Memory with declared variables

The execution of statements is realized by applying the functor Q in particular steps. First, two input statements are performed:

$$\begin{aligned} Q(\mathit{config}_2) &= [\![\mathit{read}]\!](\mathit{config}_2) = \mathit{config}_3 \\ &= ([\![S_2; S_3; S_4]\!], m_3, (\mathbf{5}), \varepsilon), \\ Q(\mathit{config}_3) &= [\![\mathit{read}]\!](\mathit{config}_3) = \mathit{config}_4 \\ &= ([\![S_3; S_4]\!], m_4, \varepsilon, \varepsilon), \end{aligned}$$

and memory after performing these two steps is depicted in Figure 5.

m_3		
x	1	**3**
y	1	$\perp$

m_4		
x	1	**3**
y	1	**5**

Figure 5. Memory after user inputs

Next, the conditional statement is executed,

$$\begin{aligned} Q(\mathit{config}_4) &= [\![\mathit{next}]\!](\mathit{config}_4) = \mathit{config}_5 \\ &= ([\![\texttt{begin var } z; z := x; x := y; y := z \texttt{ end}; S_4]\!], m_4, \varepsilon, \varepsilon), \end{aligned}$$

and a Boolean condition is evaluated

$$[\![x \leq y]\!]m_4 = \mathbf{true}.$$

Because the condition is evaluated to true, the next step is an inner block.

$$\begin{aligned} Q(\mathit{config}_5) &= [\![\mathit{next}]\!](\mathit{config}_5) = \mathit{config}_6 \\ &= ([\![\texttt{var } z; z := x; x := y; y := z \texttt{ end}; S_4]\!], m_5, \varepsilon, \varepsilon), \end{aligned}$$

and an actual memory m_5 contains also information about entering the local block (Figure 6).

m_5		
x	1	**3**
y	1	**5**
`begin`	2	$\perp$

Figure 6. Memory after entering the local block

The next step is an elaboration of a declaration inside the block:

$$\begin{aligned} Q(\mathit{config}_6) &= [\![\mathit{next}]\!](\mathit{config}_6) = \mathit{config}_7 \\ &= ([\![z := x; x := y; y := z \texttt{ end}; S_4]\!], m_6, \varepsilon, \varepsilon), \end{aligned}$$

and an actual memory m_6 is in Figure 7.

m_6		
x	1	**3**
y	1	**5**
`begin`	2	$\perp$
z	2	$\perp$

Figure 7. Memory after local declaration inside the block

The next three steps represent performing three variables assignments:

$$\begin{aligned} Q(\mathit{config}_7) &= [\![\mathit{next}]\!](\mathit{config}_7) = \mathit{config}_8 \\ &= ([\![x := y; y := z \ \texttt{end}; S_4]\!], m_7, \varepsilon, \varepsilon), \\ Q(\mathit{config}_8) &= [\![\mathit{next}]\!](\mathit{config}_8) = \mathit{config}_9 \\ &= ([\![y := z \ \texttt{end}; S_4]\!], m_8, \varepsilon, \varepsilon), \\ Q(\mathit{config}_9) &= [\![\mathit{next}]\!](\mathit{config}_9) = \mathit{config}_{10} \\ &= ([\![\texttt{end}; S_4]\!], m_9, \varepsilon, \varepsilon), \end{aligned}$$

and particular changes of memory are depicted in Figure 8.

m_7		
x	1	**3**
y	1	**5**
`begin`	2	$\perp$
z	2	**3**

m_8		
x	1	**5**
y	1	**5**
`begin`	2	$\perp$
z	2	**3**

m_9		
x	1	**5**
y	1	**3**
`begin`	2	$\perp$
z	2	**3**

Figure 8. Memory after variables assignments

After those statements, the execution of a local block must be finished:

$$\begin{aligned} Q(\mathit{config}_{10}) &= [\![\mathit{next}]\!](\mathit{config}_{10}) = \mathit{config}_{11} \\ &= ([\![S_4]\!], [\![\mathit{end}]\!]m_9, \varepsilon, \varepsilon) \end{aligned}$$

where $[\![\texttt{end}]\!]m_9 = m_{10}$ and actual memory after deleting the record of the block is in Figure 9.

m_{10}		
x	1	**5**
y	1	**3**

Figure 9. Memory after deleting the local declarations

The last step is a performing of an output statement which provides user output of computed value:

$$\begin{aligned} Q(\mathit{config}_{11}) &= [\![\mathit{print}]\!](\mathit{config}_{11}) = \mathit{config}_{12} \\ &= (\mathbf{5}, (\varepsilon, m_{10}, \varepsilon, (\mathbf{5}))). \end{aligned}$$

Our simple program is executed in particular steps by applying the endofunctor Q. These steps form a finite path in the category $\mathscr{C}\mathit{onfig}$ as we can see in Figure 10.

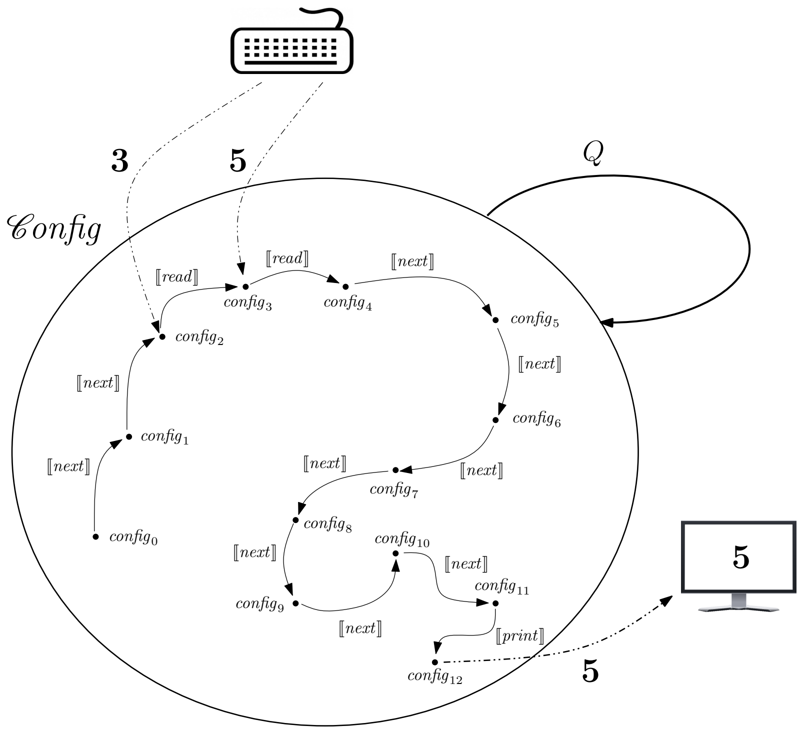


Figure 10. A program execution in coalgebra

10 CONCLUSION

Operational semantics can provide a useful information to programmers on how a program is executed, i.e. on its behavior. This information is important in the process of preparing the program or for implementation purposes. Unfortunately, traditional methods (only minor exceptions) consider some constructions as irrelevant for operational semantics. Therefore, it is hard to provide an operational semantics for a program written in a real imperative language. In this paper, we present a new approach that overcomes the lack of traditional approach. We define a simple imperative language *E-Jane* that contains most of the obvious constructs of imperative programming languages. We construct the operational semantics of this language as a coalgebra over a category of configurations. Our definition of configurations and their choice for the states, instead of memory abstractions, enables us to treat statements and declarations in a uniform way that is a further advantage of our approach. Each step of an execution is described as an application of a polynomial endofunctor Q over the category of configurations that characterizes this kind of systems. Our coalgebra also describes how input and output values go into and go out of a system. Another advantage of our approach is its possibility to get a graphical representation of the particular steps of the execution of a program which is more understandable also for practical programmers.

In this paper, we use the language *E-Jane* that has some simplifications for accentuating the principles of our approach. The constructed coalgebra can serve also as a basis for our further research. We would like to introduce also other types of values into our coalgebra and to define an operational semantics for procedures. This would be a starting point to define a coalgebraic semantics for component-based systems.

Acknowledgment

This work has been supported within the scope of the project "Semantic Technologies for Computer Science Education (SemTech)" by the Slovak Research and Development Agency under the contract No. SK-AT-2017-0012 respectively by the Austrian OeAD under the WTZ contract SK 14/2018, as well as by the Linz Institute of Technology (LIT), Project LOGTECHEDU "Logic Technology for Computer Science Education".

REFERENCES

[1] Adámek, J.—Herrlich, H.—Strecker, G.: Abstract and Concrete Categories. 1st Edition, J. Wiley and Sons, Inc., 1990.

[2] Barr, M.—Wells, C.: Category Theory for Computing Science. 2nd Edition. Prentice Hall, 1995.

[3] Crole, R. L.—Gordon, A. D.: Relating Operational and Denotational Semantics for Input/Output Effects. Mathematical Structures in Computer Science, Vol. 9, 1999, No. 2, pp. 125–158, doi: 10.1017/S0960129598002709.

[4] Ehrig, H.—Mahr, B.: Fundamentals of Algebraic Specification 1: Equations and Initial Semantics. 1st Edition. Springer, 1985, doi: 10.1007/978-3-642-69962-7.

[5] Wright, A. K.—Felleisen, M.: A Syntactic Approach to Type Soundness. Information and Computation, Vol. 115, 1994, No. 1, pp. 38–94, doi: 10.1006/inco.1994.1093.

[6] Felleisen, M.—Findler, R. B.—Flatt, M.: Semantics Engineering with PLT Redex. The MIT Press, Cambridge, MA, 2009.

[7] Fernández, M.: Programming Languages and Operational Semantics: A Concise Overview. Springer, 2014.

[8] Gordon, A. D.: An Operational Semantics for I/O in Lazy Functional Languages. Proceedings of the Conference on Functional Programming Languages and Computer Architecture (FPCA '93), Copenhagen, Denmark, 1993, pp. 136–145, doi: 10.1145/165180.165199.

[9] Hoare, C. A. R.: An Axiomatic Basis for Computer Programming. Communication of the ACM, Vol. 12, 1969, No. 10, pp. 576–580, doi: 10.1145/363235.363259.

[10] Hyland, J.—Ong, C.-H. L.: On Full Abstraction for PCF I., II., III. Information and Computation, Vol. 163, 2000, No. 2, pp. 285–408, doi: 10.1006/inco.2000.2917.

[11] Japaridze, G.: In the Beginning Was Game Semantics. In: Majer, O., Pietarinen, A. V., Tulenheimo, T. (Eds.): Games: Unifying Logic, Language, and Philosophy. Springer, Dordrecht, Logic, Epistemology, and the Unity of Science, Vol. 15, 2009, pp. 249–350, doi: 10.1007/978-1-4020-9374-6_11.

[12] Jaskelioff, M.—Ghani, N.—Hutton, G.: Modularity and Implementation of Mathematical Operational Semantics. Electronic Notes in Theoretical Computer Science, Vol. 229, 2011, No. 5, pp. 75–95, doi: 10.1016/j.entcs.2011.02.017.

[13] Kahn, G.: Natural Semantics. In: Brandenburg, F. J., Vidal-Naquet, G., Wirsing, M. (Eds.): 4th Annual Symposium on Theoretical Aspects of Computer Sciences (STACS '87). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 247, 1987, pp. 22–39, doi: 10.1007/BFb0039592.

[14] Klein, C.—McCarthy, J.—Jaconette, S.—Findler, R. B.: A Semantics for Context-Sensitive Reduction Semantics. In: Yang, H. (Ed.): Programming Languages and Systems (APLAS 2011). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7078, 2011, pp. 369-383, doi: 10.1007/978-3-642-25318-8_27.

[15] Kock, J.: Notes on Polynomial Functors. Technical report, Universitat Autonoma de Barcelona, Spain, 2007.

[16] Kuśmierek, J.—Bono, V.: Big-Step Operational Semantics Revisited. Fundamenta Informatica, Vol. 103, 2010, No. 1-4, pp. 137–172, doi: 10.3233/FI-2010-323.

[17] Kryvolap, A.—Nikitchenko, M.—Schreiner, W.: Extending Floyd-Hoare Logic for Partial Pre- and Postconditions. In: Ermolayev, V., Mayr, H. C., Nikitchenko, M., Spivakovsky, A., Zholtkevych, G. (Eds.): Information and Communication Technologies in Education, Research, and Industrial Applications (ICTERI

2013). Springer, Cham, Communications in Computer and Information Science, Vol. 412, 2013, pp. 355–378, doi: 10.1007/978-3-319-03998-5_18.

[18] Mihályi, D.—Lukáč, M.—Novitzká, V.: Categorical Semantics of Reference Data Type. Acta Electrotechnica et Informatica, Vol. 13, 2013, No. 4, pp. 64–69, doi: 10.15546/aeei-2013-0051.

[19] Mihályi, D.—Novitzká, V.: Towards the Knowledge in Coalgebraic Model of IDS. Computing and Informatics, Vol. 33, 2014, No. 1, pp. 61–78.

[20] Mosses, P.: Action Semantics. Technical report, Cambridge Tracts in Theoretical Computer Science, 1992, doi: 10.1017/CBO9780511569869.

[21] Nielson, H. R.—Nielson, F.: Semantics with Applications: An Appetizer. Undergraduate Topics in Computer Science, Springer, 2007.

[22] Novitzká, V.: Logical Reasoning about Programming of Mathematical Machines. Acta Electrotechnica et Informatica, Vol. 5, 2005, No. 3, pp. 50–55.

[23] Novitzká, V.—Mihályi, D.—Slodičák, V.: Linear Logical Reasoning on Programming. Acta Electrotechnica et Informatica, Vol. 6, 2006, No. 3, pp. 34–39.

[24] Plotkin, G.: A Structural Approach to Operational Semantics. Technical report, Computer Science Department, Aarhus University, 1981.

[25] Plotkin, G.: The Origins of Structural Operational Semantics. The Journal of Logic and Algebraic Programming, Vol. 60–61, 2004, pp. 3–15, doi: 10.1016/j.jlap.2004.03.009.

[26] Roşu, G.: CS322 Fall 2003: Programming Language Design. Lecture Notes. Technical Report, UIUCDCS-R-2003-2897, Department of Computer Science, University of Illinois at Urbana-Champaign, Lecture Notes of a Course Taught at UIUC, December 2003.

[27] Roşu, G.—Şerbănută, T. F.: An Overview of the K Semantic Framework. The Journal of Logic and Algebraic Programming, Vol. 79, 2010, No. 6, pp. 397–434, doi: 10.1016/j.jlap.2010.03.012.

[28] Roşu, G.: K – A Semantic Framework for Programming Languages and Formal Analysis Tools. In: Pretschner, A., Peled, D., Hutzelmann, T. (Eds.): Dependable Software Systems Engineering. IOS Press, NATO Science for Peace and Security Series D: Information and Communication Security, Vol. 50, 2017, pp. 186–206, doi: 10.3233/978-1-61499-810-5-186.

[29] Rutten, J. J. M. M.: Universal Coalgebra: A Theory of Systems. Theoretical Computer Science, Vol. 249, 2000, No. 1, pp. 3–80, doi: 10.1016/S0304-3975(00)00056-6.

[30] Schmidt, D.: Denotational Semantics: A Methodology for Language Development. William C. Brown Publishers, Dubuque, IA, USA, 1986.

[31] Schmidt, D.: The Structure of Typed Programming Languages, MIT Press, Cambridge, MA, USA, 1994.

[32] Schmidt, D.: Abstract Interpretation of Small-Step Semantics. In: Dam, M. (Ed.): Analysis and Verification of Multiple-Agent Languages (LOMAPS 1996). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 1192, 1996, pp. 76–99, doi: 10.1007/3-540-62503-8_4.

[33] Schreiner, W.: Computer-Assisted Program Reasoning Based on a Relational Semantics of Programs. In: Quaresma, P., Back, R.-J. (Eds.): THedu '11, Wrocław,

Poland, July 31, 2011. Electronic Proceedings in Theoretical Computer Science (EPTCS), Vol. 79, 2012, pp. 124–142, doi: 10.4204/EPTCS.79.8.

[34] Steingartner, W.—Novitzká, V.—Bačíková, M.—Korečko, Š.: New Approach to Categorical Semantics for Procedural Languages. Computing and Informatics, Vol. 36, 2017, No. 6, pp. 1385–1415, doi: 10.4149/cai_2017_6_1385.

[35] Slodičák, V.—Macko, P.: Some New Approaches in Functional Programming Using Algebras and Coalgebras. Electronic Notes in Theoretical Computer Science, Vol. 279, 2011, No. 3, pp. 41–62, doi: 10.1016/j.entcs.2011.11.037.

[36] Stoy, J.: Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory. MIT Press, Cambridge, MA, USA, 1977.

[37] Szymoniak, S.—Siedlecka-Lamch, O.—Kurkowski, M.: SAT-Based Verification of NSPK Protocol Including Delays in the Network. 2017 IEEE 14th International Scientific Conference on Informatics (Informatics 2017), IEEE, 2017, pp. 388–393, doi: 10.1109/INFORMATICS.2017.8327280.

[38] Turi, D.—Plotkin, G.: Towards a Mathematical Operational Semantics. Proceedings of Twelfth Annual IEEE Symposium on Logic in Computer Science, 1997, pp. 280–291, doi: 10.1109/LICS.1997.614955.

William Steingartner is Assistant Professor of informatics at the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics, Technical University of Košice, Slovakia. He defended his Ph.D. thesis "The Role of Toposes in Informatics" in 2008. His main fields of research are semantics of programming languages, category theory, compilers, data structures and recursion theory. He also works with software engineering and business intelligence.

Valerie Novitzká is Full Professor of informatics at the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics, Technical University of Košice, Slovakia. Her fields of research include semantics of programming languages, non-classical logical systems and their applications in computing science. She also works with type theory and behavioural modeling of large program systems based on categories.

Wolfgang Schreiner is Associate Professor at the Research Institute for Symbolic Computation (RISC) of the Johannes Kepler University Linz, Austria. His research fields include formal methods in computer science, formal semantics of programming languages, and parallel and distributed computing. He is the main developer of the formal modeling and verification systems RISCAL, RISC ProgramExplorer, and RISC ProofNavigator.

Computing and Informatics, Vol. 38, 2019, 1210–1236, doi: 10.31577/cai_2019_5_1210

CASE STUDY ON HUMAN-ROBOT INTERACTION OF THE REMOTE-CONTROLLED SERVICE ROBOT FOR ELDERLY AND DISABLED CARE

Nayden Chivarov, Denis Chikurtev

Robotics Lab, European Polytechnical University
23 Sv. Kiril i Metodiy Str., 2300 Pernik, Bulgaria
&
Institute of Information and Communication Technologies
Bulgarian Academy of Sciences
Acad. G. Bonchev St., Block 25A, 1113 Sofia, Bulgaria
e-mail: nshivarov@iinf.bas.bg, denis@iinf.bas.bg

Stefan Chivarov

Institute of Mechanics and Mechatronics – IHRT, TU Vienna
Favoritenstrasse 9-11/E325 A4, A–1040 Vienna, Austria
e-mail: e11743488@student.tuwien.ac.at

Matus Pleva, Stanislav Ondas, Jozef Juhar

Department of Electronics and Multimedia Communications
FEI, Technical University of Košice
Park Komenského 13, 041 20 Košice, Slovak Republic
e-mail: {matus.pleva, stanislav.ondas, jozef.juhar}@tuke.sk

Kaloyan Yovchev

Robotics Lab, European Polytechnical University
23 Sv. Kiril i Metodiy Str., 2300 Pernik, Bulgaria
&
Faculty of Mathematics and Informatics, Sofia University
5 James Bourchier Blvd., 1164 Sofia, Bulgaria
e-mail: k.yovchev@fmi.uni-sofia.bg

Abstract. The tendency of continuous aging of the population and the increasing number of people with mobility difficulties leads to increased research in the field of Assistive Service Robotics. These robots can help with daily life tasks such as reminding to take medications, serving food and drinks, controlling home appliances and even monitoring health status. When talking about assisting people in their homes, it should be noted that they will, most of the time, have to communicate with the robot themselves and be able to manage it so that they can get the most out of the robot's services. This research is focused on different methods of remote control of a mobile robot equipped with robotic manipulator. The research investigates in detail methods based on control via gestures, voice commands, and web-based graphical user interface. The capabilities of these methods for Human-Robot Interaction (HRI) have been explored in terms of usability. In this paper, we introduce a new version of the robot Robco 19, new leap motion sensor control of the robot and a new multi-channel control system. The paper presents methodology for performing the HRI experiments from human perception and summarizes the results in applications of the investigated remote control methods in real life scenarios.

Keywords: Service robots, human-robot interaction, remote control, speech recognition, gesture recognition, ROS

Mathematics Subject Classification 2010: 68-T40

1 INTRODUCTION

Prognoses of the United Nations show that there is a worldwide trend of continuously aging population and respectively increasing number of people with mobility difficulties [1]. Most of the elderly and disabled citizens want to live in their own houses [2] using the new smart home technologies for as long as possible, thus a robot will have to perform real-life interaction with them [3, 4]. The proposed remote-controlled service robot for elderly and disabled care can help them with tasks of the every day life such as reminding them to take medications, serving food and drinks, turning on and off electronic devices, alerting when the user's health status is getting worse and connecting them with their physician, relatives or an emergency ambulance.

Assistive service robotics now expands as an alternative for improving the quality of life of elderly and disabled [5, 6]. From the data of the International Federation of Robotics it could be expected that in the period of 2019–2021 about 39 million of new service robots for personal use will be produced and about 34 000 robots for the support of elderly and handicap assistance will be installed [7].

Remote control is widely used in many applications. The basic concept is that the robot should replace humans where people are exposed to unfavorable conditions or performing routine day-to-day activities [8, 9]. The essence of this robot management method is that the person controls the robot remotely, and the robot must possess the necessary qualities and functions to perform successfully in the intended tasks [10].

When talking about assisting people in their homes, it should be noted that they will, most of the time, have to communicate with the robot themselves and be able to manage it so that they can get the most out of the services the robot provides [11].

The aim of this study is to investigate the control methods for remote human-robot interaction. The paper is structured as follows. Section 2 describes the hardware, software and control system of our robot "ROBCO 19". In Section 3, different methods for remote control of the robot are thoroughly investigated. The capabilities of these methods for human-robot interaction have been explored both in terms of utility and functionality. Section 4 presents the methodology for performing human perception experiments with the robot and the application of proposed remote control methods in real life scenarios. Section 5 describes the performed test scenarios and summarizes the results from the real life experiments.

2 ASSISTIVE SERVICE ROBOT – ROBCO 19

ROBCO 19 is the next iteration of the personal assistive robot "ROBCO 18" from 2018 [12]. The robot was redesigned as follows. The laser scanner was replaced with a new RPLidar scanner which allows outdoor navigation. The sensor system was upgraded by adding an Intel RealSense camera for improved object recognition and manipulation. The microcontrollers have been upgraded with Teensy hardware. Also, a new graphical user interface and new software have been developed for autonomous navigation, collision avoidance and execution of predefined tasks. New electro-actuating systems (high power DC motor drivers), batteries and recharging docking system to allow 24/7 service were installed. As a result, we have a remote-controlled service robot for elderly and disabled care "ROBCO 19" (Figure 1).

The hardware components are described by their location in the layers of the mobile platform. In the first layer are DC motors, encoders, distance sensors, orientation sensor, batteries, controller and drivers. In the second layer are the computer of the robot and the laser scanner RPLidar. In the third layer are the articulated arm Mover4 and the RealSense camera.

Figure 2 shows the connections between robot devices. The robot's computer is connected to the mobile base controller, arm controller, RealSense and RPLidar over USB communication ports. The computer is the main computational device of the robot – sends/receives data, computes data, runs algorithms and controls the robot.

RPLidar is used for the autonomous navigation of the robot [13, 14]. The Lidar scans the area around the robot in 2D. The scanned parameters are described in Table 1. Thanks to the Lidar, the navigation system easily finds the robot's location and navigates smoothly and safely [15].

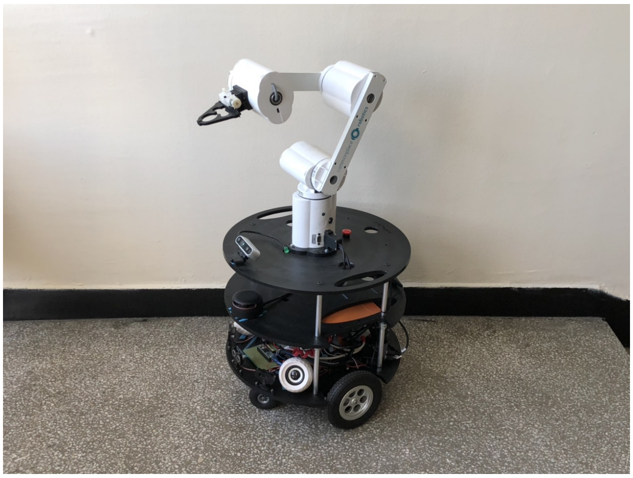


Figure 1. The service robot ROBCO 19

The RealSense camera is used for the robot's vision system. When we combine its depth sensor with its camera, we can recognize objects and locate their position in 3D space [16]. This property is very important for autonomous grasping of objects, for human recognition, and for autonomous navigation [17].

The Teensy controller works also on a separate level. It reads and converts the signals from the encoders, distance sensors and MPU (Motion Processing Unit),

Device	**Description/Characteristics**
Computer	Intel Xeon E3-1230 v5, 8 GB RAM
Mobile base controller	Teensy 5.2, based on the MK20DX256 32-bit ARMCortex-M4 and 72 MHz CPU
Arm controller	PCAN-USB
RealSense camera	Depth Stream Resolution and FpS: 1 280 × 720, 90 fps, Depth Distance: Min: 0.1 m, Max: 10 m, RGB Resolution and FpS: 1 920 × 1 080 at 30 fps.
RPLidar scanner	Distance Range: 25 meters, Sample Rate: 16 000 fps, Scan Rate: 15 Hz (adjustable between 10–20 Hz), Angular Resolution: 0.3375 degrees, Communication Speed: 256 kbps.
MPU – Motion Processing Unit	MPU-9250 Nine-Axis (Gyro + Accelerometer + Compass) MEMS Motion Tracking Device
Drivers and motors	12 V – 10 A drivers, 12 V DC motors with gears
Mover4 robotic arm	Commonplace Robotics GmbH, four degrees of freedom, planar kinematic structure

Table 1. Specific characteristics of the robot's hardware components

then sends the data to the PC. The controller receives data back from the robot's computer for running the platform, i.e. controlling the wheels. Then Teensy sends control signals to the drivers using Pulse Width Modulation (PWM).

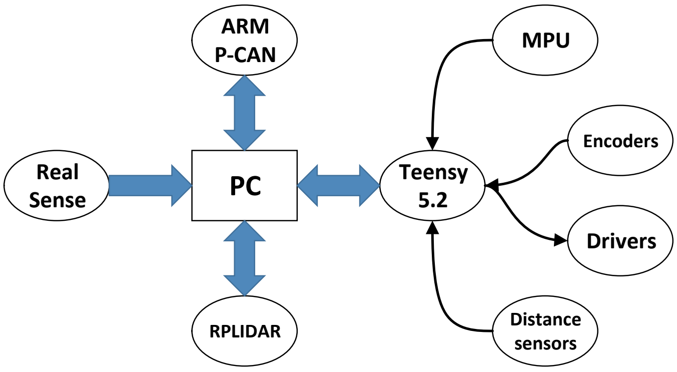


Figure 2. Hardware system of the robot

2.1 Mobile Robot Platform

A differential drive robot is a wheeled robot with two controllable wheels, as shown in Figure 3. To maneuver any differential drive robot on a plane, the robot needs a linear velocity V and a heading θ. By controlling the velocity and orientation, the path of the robot can be planned.

While we can vary the velocity of each wheel, for the robot to perform a side turn, the platform must rotate about a point that lies along the left and right wheels' common axis. The point that the robot rotates about is known as the ICC – Instantaneous Center of Curvature.

By varying the velocities of the two wheels, we can vary the trajectories that the robot takes. Because the rate of rotation ω about the ICC must be the same for both wheels, we can write the following equations:

$$\omega\left(R+\frac{l}{2}\right)=V_r,$$

$$\omega\left(R-\frac{l}{2}\right)=V_l$$

where l is the distance between the centers of the two wheels, V_r, V_l are the right and left wheel velocities along the ground, and R is the signed distance from the ICC to the midpoint between the wheels. At any instance in time we can solve for R and ω:

$$R=\frac{l}{2}\frac{V_l+V_r}{V_r-V_l}; \quad \omega=\frac{V_r-V_l}{l}.$$

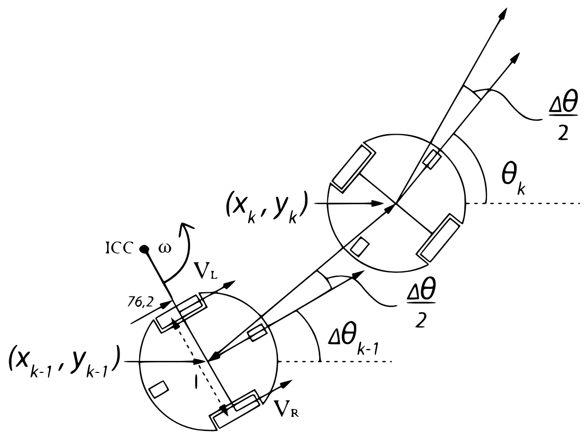


Figure 3. Differential drive mobile platform

There are three interesting cases with this kind of drive:

1. If $V_l = V_r$, then we have a linear forward motion in a straight line. R becomes infinite, and there is effectively no rotation – ω is zero.
2. If $V_l = -V_r$, then $R = 0$, and we have rotation about the midpoint of the wheel axis – we rotate in place.
3. If $V_l = 0$, then we have rotation about the left wheel. In this case $R = \frac{l}{2}$. The same is true if $V_r = 0$.

In Figure 3, assume the robot is at some position $(x;\ y)$, headed in a direction forming an angle θ with the X axis. We assume the robot's center is at a point in the middle of the wheel axle. By manipulating the control parameters V_l; V_r, we can get the robot to move to different positions and orientations. Please note that V_l and V_r are wheel velocities along the ground.

Knowing velocities V_l and V_r we can find the ICC's location using: $ICC = [x - R\sin\theta, y - R\cos\theta]$ and at time $t + \delta t$ the robot's position will be:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega\delta t) & -\sin(\omega\delta t) & 0 \\ \sin(\omega\delta t) & \cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega\delta t \end{bmatrix}.$$

This equation simply describes the motion of the robot rotating at a distance R about its ICC with an angular velocity of ω.

2.2 Articulated Robotic Arm

The MOVER 4 robot of Commonplace Robotics GmbH has four degrees of freedom and planar kinematic structure [18]. The Table 2 lists the kinematic parameter values of the MOVER 4. Three of the degrees of mobility provide the positioning, and the fourth orientates the End Effector (EE) relative to the Z-axis of the coordinate system associated with the base of the robot. The coordinates of any point in the kinematic chain, and in particular the EE, can be derived from geometric considerations.

Joint Number **n**	Constraints θ_n **[DEG]**	Length L_n **[m]**	Twist Angle α_n **[DEG]**	Offset S_n **[m]**
1	−150, 150	0	90	0.206
2	−50, 65	0.19	0	0
3	−110, 140	0.22	0	0
4	−140, 135	0.095	0	0

Table 2. Kinematic parameters

If $q1$, $q2$, $q3$ and $q4$ are the generalized state space coordinates of the robotic manipulator, then the equations for the (X, Y, Z)-coordinates of the EE are as follows:

$$X = \cos(q_1).(\cos(q_2).L_2 + \cos(q_2+q_3).L_3 + \cos(q_2+q_3+q_4).L_4),$$
$$Y = \sin(q_1).(\cos(q_2).L_2 + \cos(q_2+q_3).L_3 + \cos(q_2+q_3+q_4).L_4),$$
$$Z = S_1 + \sin(q_2).L_2 + \sin(q_2+q_3).L_3 + \sin(q_2+q_3+q_4).L_4.$$

2.3 Control Software

ROS is a meta-operation system for robot control [19]. ROS provides access to a number of open source packages that provide various applications and features [20, 21]. This section describes the properties and features of some of the ROS packages that we use to control our service robot. The following packages are presented: SLAM, MoveIt, Robot node, ROS-bridge, Web_sockets, RPLidar, LeapMotion node, and Kinect node. Each package performs specific functions according to its purpose, and all nodes are connected to each other via the ROS Master.

Robot_node is the main node for controlling the robot. It calculates the specific parameters for each robot. All other nodes are connected to this node. For the robot described in this article, this package performs the following functions: reading the robot model, calculating speed (accelerometer), reading data from all sensors, reading incoming data from other nodes, and sending control data to the control engine (Teensy) of the mobile platform and the arm.

The code of this package describes the characteristics of the robot's mobile platform such as wheel diameter, platform width and length, encoder resolution, mobile platform type (two/four-wheel drive, differential/Mecanum/Omni drive), sensor location, drive controllers, and others. The diagram of connections between the nodes and their functions is in Figure 4.

ROBCO 19 has three main operation modes:

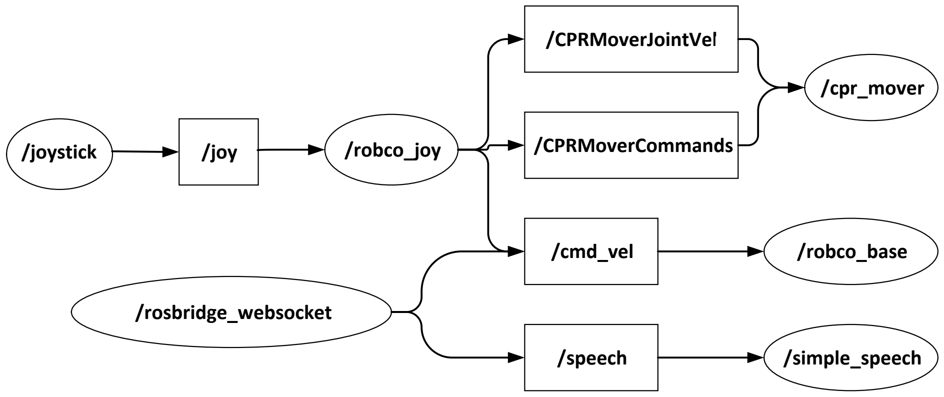


Figure 4. Node connection diagram

Manual Mode: In this mode, the robot receives commands from the user through the available control methods. All algorithms and programs for automated tasks are stopped. The user has full control of the robot motions but has to do all the command work to perform a task. ROBCO 19 has a user-friendly web-based User Interface for remote control [31, 32]. The robot can be controlled either by joystick, tablet/computer/phone (via WEB interface), voice or gestures.

Semi-Autonomous Mode: This mode combines Manual Mode with algorithms and programs for automated functions. Some of the tasks are predefined and the user only has to choose which task to be performed. FlexBe behavior engine's user interface is used for this purpose, which allows executing high-level tasks while the operator is able to influence the execution during runtime.

Autonomous Mode: The Autonomous mode requires only a single command from the user, then the robot performs all the necessary tasks until the goal is completed. It also uses FlexBe behavior engine which allows for fully autonomous execution of tasks and behaviors.

FlexBE is a powerful and user-friendly high-level behavior engine for generating complex robot behaviors without the need to manually code them. Among its basic capabilities, which interface standard functionality or your own system-specific features, state machines can be easily composed via the provided drag & drop editor [33].

To switch between operating modes we have added special buttons in the Web User Interface. This makes it very easy for the user or operator to change modes.

3 METHODS FOR HUMAN-ROBOT INTERACTION

Because we investigate human-robot interaction, various methods of controlling a service robot have been developed and explored. Therefore, the creation of a multi-channel (multi-modal) architecture has to be established to allow the robot to be controlled by all different methods at the same time [22]. In order to avoid conflicts at this stage, the structure itself gives each method a certain priority. The remote control channels – hardware and software joystick – have the highest priority. Following are the channels of control by gestures and voice. With the lowest priority of the direct-control channels is the Web-based control. Channel prioritization is described in detail in [12]. This architecture is easy applicable and re-configurable when working with ROS because ROS works on the principle of Internet protocols. Thus, each method publishes commands through a different channel, but to the same subscriber. The multi-channel system monitors and manages data traffic.

3.1 Gesture Control via Kinect Sensor

In order to achieve convenience for different robot users, we have developed remote control methods through different devices. Gesture control is intended to replace standard voice commands because some elderly and disabled people have speech defects or are mute/deaf. Human-robot interaction through gestures is a good alternative and also helps maintain and develop motor skills of the users [23].

Control of the robot by recognition of hand gestures was implemented using the Kinect sensor.

Kinect is a sensor consisting of an RGB camera and a depth sensor. It provides functions to recognize the human skeleton and to monitor the positions of the joints of the whole body[1]. We have developed an algorithm and a program to process the data from human hands, and with processing we can control the robot's mobile platform or its articulated arm.

A particular feature of this type of control is that the user gives commands to the robot through a separate computer. The Kinect is connected to the user's computer and the user remotely controls the robot with hand movements.

The essence of this method is the following: Once the user is in front of the Kinect and all the joints of the hands and the head are recognized, the program starts working. As feedback for the user, the program displays specific words in the window menu: Working, Forward, Stop and others. The principle of operation, different zones that we have set empirically, and on which part the control is applied are shown in Figure 5.

Actually, there are two modes of operation: sitting or standing in front of the Kinect sensor. The standing one was not preferred by the users, but it could be used for rehabilitation purposes.

To identify the different directions for the mobile platform, we have identified nine hand positioning zones. The inactive/passive zone is in the body area so that it covers the natural position of the hands when we are seated. When the hands fall

[1] You can see all the skeleton points in Figure 2 of [24] article `https://www.sciencepubco.com/index.php/ijet/article/download/10152/3614`

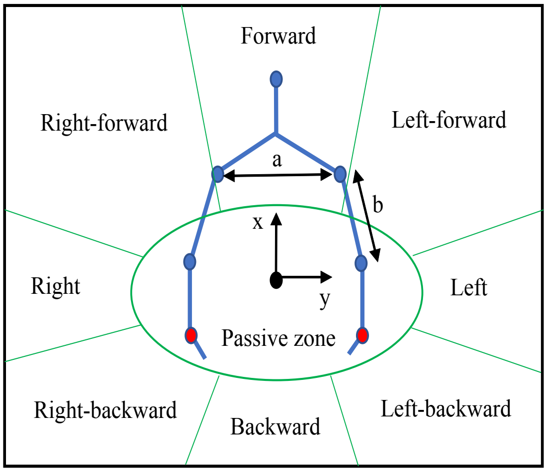


Figure 5. Control areas and dimensions of the measured parameters

into this zone, we send a stop command. When the user moves one of their hands to another zone, commands are then sent to the robot.

In the forward, backward, left and right zones, we send commands for linear motion or rotation in place, with control of the speed – as the wrist joint is closer to the inactive/passive zone, the lower is the speed. When the user's arm is in the zones like right-forward or left-backward, the control is based on the differential drive principle. The velocities change depending on the distance to the inactive/passive zone, the ratio of the linear/angular velocity depends on the ratio of the coordinates x and y.

An important feature of this method is that the program should be configured according to the mobility of the user. Then, all zones are automatically determined according to the length between the joints. Initially, the inactive/passive zone is specified. It is defined as an ellipse with a radius of x and y. Based on the parameters of the inactive/passive zone the parameters of the other zones are set. In this way, the problem of the different height of people is solved. For people with longer hands the zones have larger dimensions and vice versa.

The proportion is determined by taking half the distance between the shoulders a and adding half of the relative length of the arms from the shoulder to the elbow b to set the y radius. For determining the radius by x, only half of a is taken. The center of the global map is the center of the global coordinate system. So, the control parameters for the robot are: $v = f(x) * k$; $\omega = f(y) * m$, where k and m are proportional coefficients.

Another important problem is the simultaneous submission of commands with both hands in opposite directions. In this case, a safety algorithm is implemented. If the user gives a forward command with both hands, the lower values are taken. If one hand is only forward and the other is forward and left/right, then the angular values are added. In case of opposite commands, both left and right or back and forth, we send a "stop" command.

3.2 Gesture Control via Leap Motion Sensor

Leap Motion is a stereo camera sensor and its main task is to recognize the human hand. Its main features are recognition of the position of the palm and each individual finger in 3D coordinates [28, 29, 30]. In this way, we can recognize whether a fist is closed and how many fingers are folded or extended. All of these data can be used to control a mobile robot or articulated arm. We have the option to choose a method, criterion, or a complex set of several criteria for extracting data and converting them into control signals.

The prerequisite for running the program is to have a valid hand recognized by the sensor itself, otherwise no commands are given. To start sending commands, when the above condition is already met, the user has to close his hand into a fist first. These considerations have been made because of the need to ensure safety.

The method under development includes the following. Initially, we recognize if there is a hand over the sensor, then we take the palm coordinates and begin to convert them so that correct control signals are generated for the mobile platform and the robot's arm. Currently the control of the mobile platform and the arm is separated in two different programs. It is planned to integrate the two programs by introducing the option of switching the controlled device after a certain gesture.

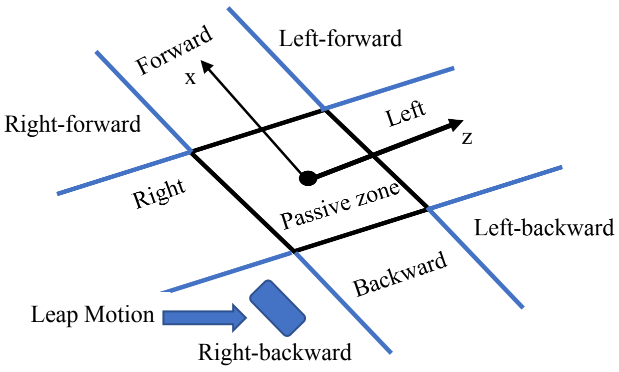


Figure 6. Working area in Cartesian coordinates for control of the mobile base using Leap Motion Sensor on the table

To control the mobile platform, linear and angular velocity have to be submitted. Since we get the values (from 0 to 300 mm) of x, y and z coordinates from the palm position output data (Figure 6), it is relatively easy to convert these values into mobile platform control data using only the data obtained from x and z. When the hand shifts out of the inactive/passive zone, we assign the linear velocity v to x, and from the value of z we assign the angular velocity ω.

When handling the anthropomorphic manipulator, we use the 3D coordinates of the palm. The goal is to position the gripper of the manipulator by computing the necessary rotation of all its joints. To achieve this goal, we initially identify the passive zone where no commands are issued. Because this time we are working in 3-dimensional space, this area is about the origin (zero) coordinate in the form of a cube with sides of 120 mm (Figure 7). As soon as the user's hand comes out of this zone, the transformed data x, y, z of the palm are sent to the arm controller. In this case, we use the x data to move forward and backward, y to move up and down, and z to move left and right. Accordingly, the combination of the three parameters gives us the final movement of the arm.

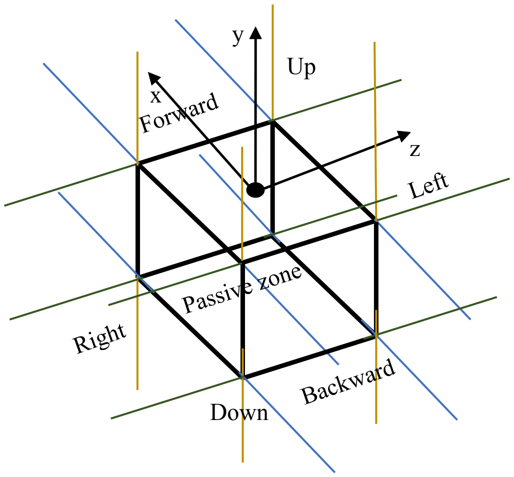


Figure 7. Working area in 3D for control of the Mover4

We multiply the values of x, y, z by a factor of 0.5 in order to transform them into the actual speeds of the motors of the arm, because the range of the Mover 4 velocities is from 0 to 150 mm/s, and the range of the leap motion is measured from 0 to 300 mm. The directions of rotation are determined by the equations for the inverse kinematics of the robot arm manipulator described in Section 2.2.

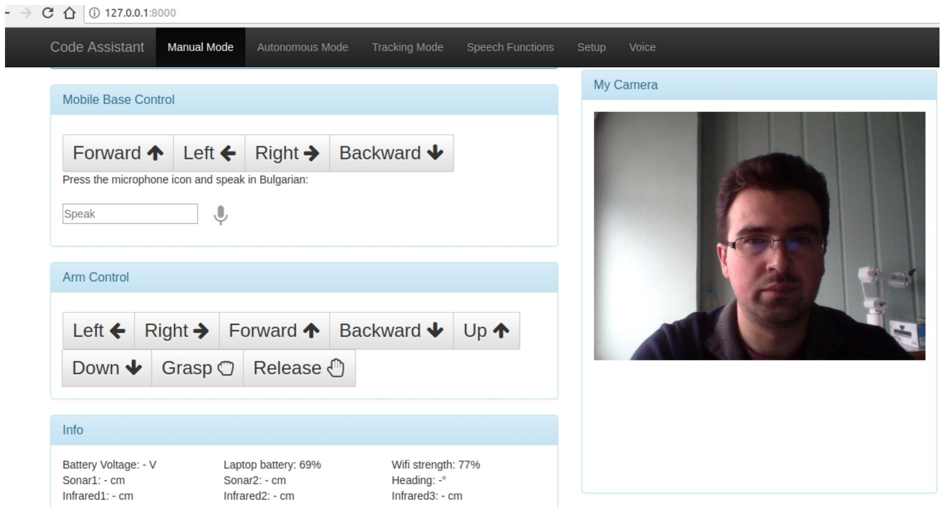
The system of equations for all parameters is:

$$\begin{aligned}
\omega_0 &= y * 0.5;\\
\omega_1 &= (x * -0.5) + (z * 0.5);\\
\omega_2 &= x * -0.5 + (z * 0.5);\\
\omega_3 &= x * 0.5 + (z * -0.5).
\end{aligned}$$

To get closer to the natural human behavior, the grasping with the articulated arm is like palm grasping. When we close the palm, the gripper closes, and when we open the palm, the gripper opens. Again, for safety reasons when a valid hand is not detected or it is within the range of the passive zone, no commands are sent.

3.3 Web User Interface

A web-based user interface has been developed and added to obtain and manage robotic system status data (Figure 8). The main advantage of web-based interfaces is the ability to use any modern device that supports TCP/IP protocol and is on the same local network with the robot. Depending on network configuration there is also the possibility to control the system over the Internet. The web-based interface has visual click/touch design, allowing it to be used regardless of the type of the device – smartphone, tablet, laptop or full-sized desktop computer.



Press the microphone icon and speak in Bulgarian:

Battery Voltage: - V
Sonar1: - cm
Infrared1: - cm
Laptop battery: 69%
Sonar2: - cm
Infrared2: - cm
Wifi strength: 77%
Heading: -°
Infrared3: - cm

Figure 8. Part of the developed Web User Interface

The web-based interface is divided into separate sections depending on their functionality. The interface provides easy and convenient control of the robot in manual, semi-autonomous and autonomous mode. Commands and control signals can be sent via virtual buttons, a virtual joystick, or voice commands. The interface also provides the capability to monitor the robotic system as well as to configure robot's every parameter.

In the interface there are added buttons to control the movement of the mobile platform. Four basic moves are pre-programmed – forward/reverse linear movement, as well as left/right turn and rotational movement. Pressing these buttons once performs the corresponding movement at a preset and adjustable distance or angle.

Buttons for manipulating the articulated robot arm manipulator are also provided. There are 8 of them together. Two of these buttons are to open and close the gripper. The other 6 buttons are for positioning the manipulator. The buttons perform upward, downward, left and right movement of the gripper, and also left and right rotation at the base of the manipulator. As with the mobile platform's buttons, the one-time press of these buttons performs motion with a preset and adjustable distance. The selected Mover4 robot manipulator has 4 degrees of freedom. This means that more than one actuator is used for the forward, backward, upward and downward movement of the gripper. For this reason, it is necessary to solve the kinematics of the manipulator and program it into the user interface.

A semi-autonomous map navigation feature has been added to the web-based user interface [34]. It is programmed as a separate section of the interface. It visualizes the map of the room created by the robot as well as the current position and orientation of the robot [35]. The UI allows the robot's current position and orientation to be kept in a specially created database at any time and named in a user-friendly way. From a convenient drop-down menu, the already saved database entries can be selected and submitted to the robot's control system. It will, then, navigate the robot all the way from the current to the assigned position.

The web-based user interface is a great feature for a robotic system which improves the overall user experience. It gives the user the comfort of controlling the robotic system directly by their favorite device instead of a separate specialized controller.

3.4 Voice Control

Speech to text conversion is used to trigger robot's different top level behaviors (FlexBe) or for manual control when in manual mode. Bulgarian speech recognition was implemented using the Google Cloud Speech-to-Text API [36]. ROBCO 19 must be connected to the Internet and the audio capture is transferred to Google servers for STT (Speech To Text) processing. The TTS (Text To Speech) synthesis in various languages is provided by the Espeak, open source software speech synthesizer. As there is no good quality synthesized Bulgarian language voice in Linux, we use VMware Virtual Windows machine to provide Bulgarian TTS, using the Windows SAPI (Speech Application Programming Interface).

For testing and simplicity we have developed predefined phrases for voice control. For example, "robot forward" means for the robot to move 0.3 m forward. Speech to text recognition is done by Google API and parsed using "word spotting". When the API returns the recognized text, we search it for the specific phrase using simple code in JavaScript. But the problem is that if you say for example "the robot goes forward" it also detects that there are the words "robot" and "forward" and it sends the commands for the movement. All these predefined phrases are for demonstration purpose, only to show that the robot can be controlled by voice commands and the Natural Language Understanding module (currently only "word spotting") will be improved for next releases.

We can add more complicated actions for example, the phrase "robot go to the kitchen" should instruct the robot to go to a specific place at home. If we saved that location in the autonomous navigation then this phrase could be directly connected to the action of performing movement to the position in the kitchen. In addition, specific voice commands may activate one or more actions from the FlexBE engine.

For the issue of security and privacy it is required for some applications not to use the third party cloud services, which could store all the voice requests for future improvement of the service. For this purpose, we tested the Julius LVCSR (Large Vocabulary Continuous Speech Recognition) engine with freely available English acoustic models[2] and fixed grammar [37]. However, for the proposed purpose of free phrase processing we will need more complex language models. Satisfactory models could be built for this purpose using freely available corpora of different languages using modern deep learning techniques for neural networks. More robust language models can be trained using a unique approach proposed in [38], where the methodology for training language models (LM) without training data is described.

The methodology relies on the iterative process, in which, at the beginning, an initial model is trained only from system vocabulary and randomly generated phrases, which summarize all devices, actions and functions of the robot. Then, the LM can be retrained using recognized phrases automatically or semi-automatically (after corrections made by a human expert). The final model then enables significantly higher number of phrases than fixed grammar. Also the detection of emotions from the recognized speech could be applied [39] for more natural human-like communication behavior.

We are currently working on a joint Slovak-Taiwan bilateral DeepSpeech project called Deep Learning for Advanced Speech Enabled Applications. Experience from using Kaldi, Tensorflow and DeepSpeech (Mozilla project with coincidentally the same acronym as our project) gives us the opportunity to build a high quality local Neural Network based speech recognition engine with higher accuracy than Julius based on finite-state transducers. For the LVCSR task, the Kaldi DNN approach achieved 8 % Word Error Rate (percentage of incorrectly recognized words out of all words in a test set) in English and 17 % in Slovak.

After combining Slovak and English using Language Identification Module the system was able to recognize English words with similar 8 % WER and Slovak words with 16 % WER. A similar approach could be used for English and Bulgarian language after gaining enough Bulgarian speech data and corpuses. Currently our Taiwanese partner from TaipeiTech is proposing a joint project with the European Polytechnical University (EPU) to work towards this goal [40].

After successful recognition of a continuous speech phrase, the module of Natural Language Understanding (NLU) needs to be in place to extract the intended meaning and Natural Language Generation (NLG) to prepare a meaningful sentence with the required response, if needed, for this application. Rule-based NLU module can be adopted from our previous solution described in [41].

[2] `https://sourceforge.net/projects/juliusmodels/`

Usage of more complex voice commands can result in occurrence of ambiguities, which can be solved in dialogue interaction. To manage dialogue in task-oriented scenarios, the VoiceXML-based dialogue manager can be integrated. We designed and developed an advanced VoiceON unit, that implements interpretation of enhanced VoiceXML scripts (see [42]). Our modification of VoiceXML enables integrating robot action control directly through VoiceXML <prompt> element. Moreover, VoiceXML integrates frame-based approach to natural language generation, which well fits the proposed scenario.

VoiceON dialogue manager supports JavaScript language and easy extension through calling own executables. Position of the robot can be obtained by calling a special object and then it can be inserted into the VoiceXML variables for further use. Variables can be stored in the application scope of the VoiceXML application and easily used to provide the position description in any state of the interaction.

4 METHODOLOGY FOR USABILITY EXPERIMENTS

The tests with elderly and disabled were performed during the project "Tele-controlled Service Robots for Increasing the Quality of Life of Elderly and Disabled, No. DN 07/23 – 15.12.2016" financed by the Bulgarian National Science Fund for the European Polytechnical University – Pernik, Bulgaria.

The tests have been conducted with 30 participants in two target groups, 15 elderly men and women and 15 disabled people using multi-channel robot management software for controlling ROBCO 19. They controlled the robot through virtual joystick, voice commands, mimic gestures and head movements. In the first project year the tests were performed with elderly people at the Scientist house of the Bulgarian Academy of Sciences (Figure 9). The elderly people involved in the experiments were between 66 and 81 years old (average 75.3 years old). For the second project year tests were performed with disabled volunteers at Union of the Disabled in Bulgaria, Trojan Branch (Figure 10). The disabled people were between 34 and 80 years old (average 63.8 years old) and with incapacity rate between 59 % and 90 % (average 78.2 %). The incapacity rate is determined by the Bulgarian National Expert Medical Commission.

There have been real tests of the proposed remote control of the mobile service robot performed with elderly and disabled people, including:

- Efficient support for the elderly and disabled at their different needs;
- Reminder when to take their medication;
- Food and beverage service;
- Switching on/off electronic devices;
- Alerting the doctor, relatives, or ambulance service with possible indicators of deterioration in health (day and night monitoring of elderly and disabled, described in details in [43]);

Figure 9. Performed tests with elderly

Figure 10. Performed tests with disabled

It is important to note that no special education or technical knowledge is required to work with the robot. The purpose of these tests was to verify the application effectiveness and the reliability of the robot systems. In the experiment room, two tables were placed at a distance of about 3 to 4 m apart. On the first table there was: the computer and the smart-phone of the experiment lecturer, the manual control joystick of ROBCO 19, the Leap Motion and the Kinect. The tests duration with each of the target groups was one day.

In the frame of four academic hours in the morning, the lecturer has presented, demonstrated and trained all methods and modes for control of the robot to each target group. After the explanation and training, the lecturer prepared the robot for the experiments. On the opposite table there were household items for the robot to pick up such as: a 330 ml plastic bottle of mineral water, a pack of biscuits, a medicine box, a small soft drink bottle and more objects weighing up to 250 g. The goal for the elderly/disabled participants was to grasp one of the items on the table, take it to the other table and deliver it to someone of the other participants in the tests, thus the three operation modes were tested and evaluated by the users. All experiments were conducted under the same conditions, with some features described for each individual experiment. The facilitator had the task of turning on the robot and all necessary devices, as well as ensuring proper operation and safety.

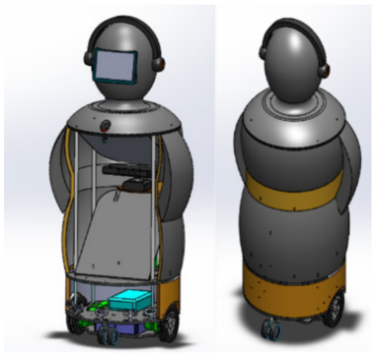
Within four hours at the afternoon, the target groups members have used all of the presented control methods and operational modes of the robot, fulfilling the described above goal of the test. Each participant has chosen the appropriate methods and modes of robot control, depending of its personal preferences and motion difficulties. Finally, the groups of the elderly and disabled completed a questionnaire about their impressions and attitudes toward the robot. In the questionnaire more than one answer per question was allowed.

The study was conducted according to the project methodology and approved by the Ethics Committee, in coordination with Physiological Department of the European Polytechnical University (EPU). Ethics Committee was composed of the project leader and scientific members. The Committee in coordination with EPU Physiological Department has approved the methodology for the tests and the included questionnaires.

Under Regulation (EC) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of individuals with regard to the processing of personal data, test subjects were informed that the personal data in the survey were not collected for commercial purposes but would be used for statistical and scientific purposes.

The questionnaire results show that both elderly and disabled people expressed a positive feedback about using remote-controlled service robot ROBCO 19 for their needs, performing real interaction with the robot. Each participant conducted the tasks, using the chosen methods before the experiments.

In terms of functionality, they expect the robot to be able to monitor their health, bring heavy or hard-to-reach items, carry food, water and drugs, and contact the


a)

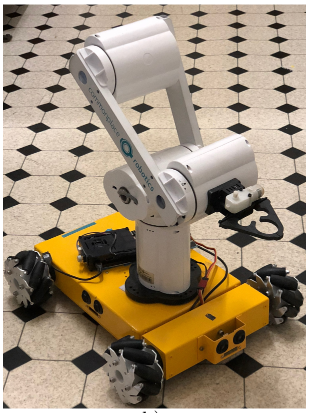

b)

Figure 11. a) Anthropomorphic (humanlike) design; b) Robco 20 – industrial design (in development with Mecanum wheels)

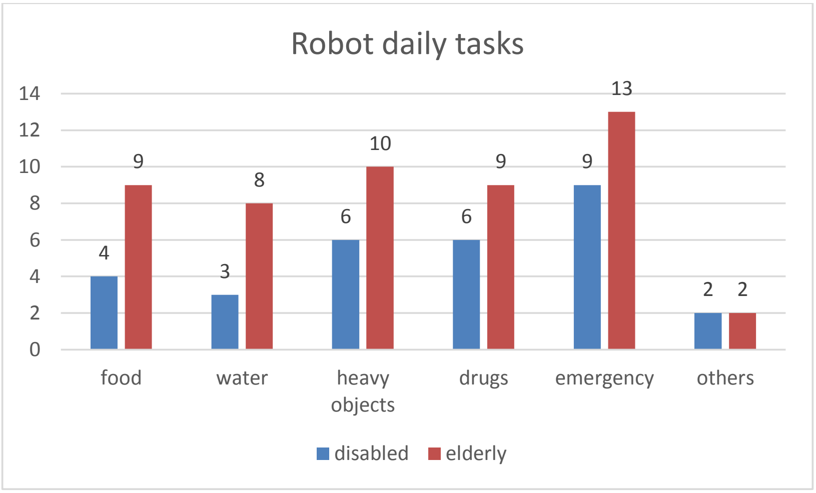


Figure 12. Robot daily tasks desired by the elderly and disabled

first aid in case of emergency (Figure 12). In the "others" section, volunteers have requested rehabilitation as an additional feature of the robot.

When asked about the industrial design, elderly participants (most of them living alone) preferred a human-like design (53 %) for the robot (Figure 11 a)), while the disabled prefer more functional one (60 % for Robco 20) (Figure 11 b)). For the control of the robot, both elderly and disabled individuals prefer voice control (Figure 13). The most effective and less time consuming control methods are physical joystick (for the elderly) and virtual joystick embedded in the Web UI (for the

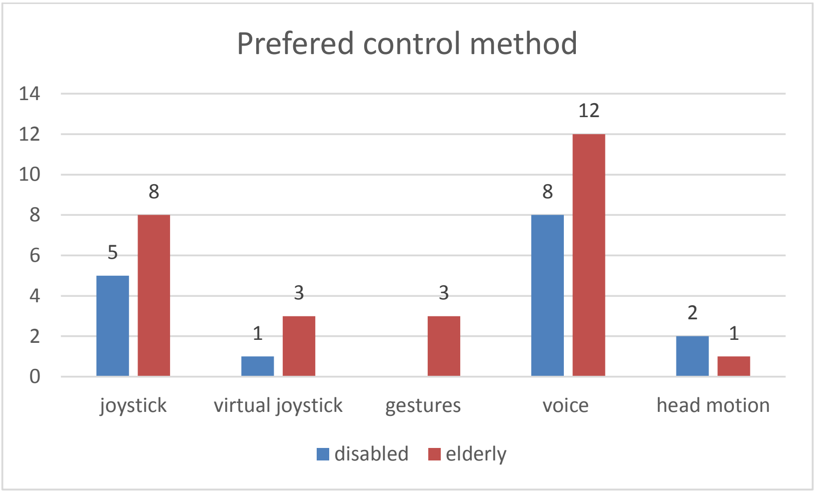


Figure 13. Preferred control method

disabled). For the voice control of the robot, they prefer the female voice (Figure 14).

Finally, elderly prefer to remote control the robot themselves, while disabled people would allow the robot to be controlled by their relatives, social caregivers and doctors as well (Figure 15). The most preferred operation mode is the manual mode, where users feels more secure and independent, while controlling the robot.

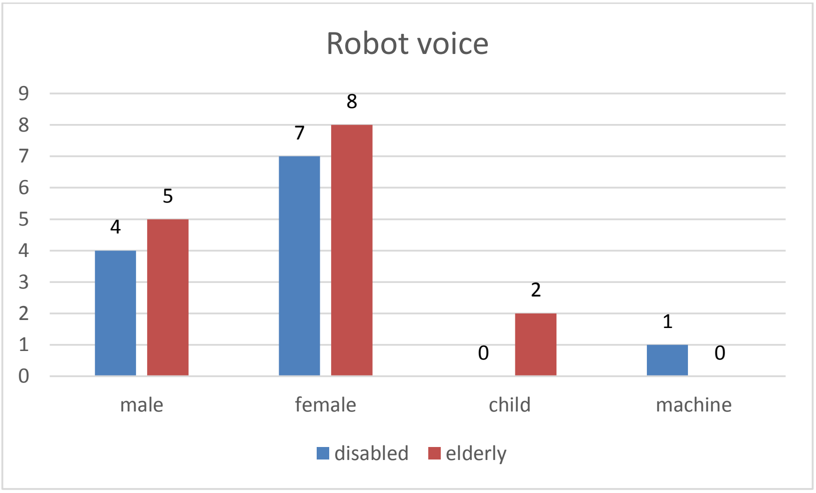


Figure 14. Preferred voice of the robot

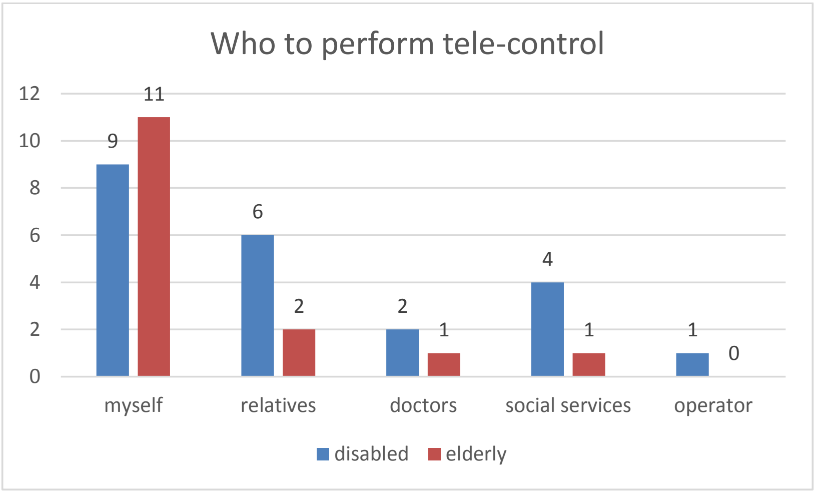


Figure 15. Preferred operator of the robot

5 CONCLUSION

This research involved a detailed investigation of different methods of remote human-robot interaction: gesture control via Kinect and Leap Motion sensor, web-based user interface and voice-based control. These methods were applied to a mobile platform robot equipped with an articulated arm.

We have obtained useful information about what elderly and disabled people consider useful for their care from our robot. Interestingly, the elderly prefer to control the robot by themselves (73 %), but disabled people would like to do the remote control by somebody else (57 %) for instance their relatives or nurses. The results show that the described methods for remote control of service robots are convenient and easy to use. Experienced participants completed all tasks using different methods. The trial period was relatively short, indicating that the methods were natural and enjoyable for such application.

The two-thirds of participants prefer to control the robot by choosing the most appropriate method and mode for them. We can conclude that both elderly and disabled prefer a female voice and the joystick control methods. Due to their motion difficulties, elderly and disabled do not prefer gesture and head motion controls. The various control methods provide the opportunity for each user to choose the method which is the most convenient for him/her. The multi-channel control system allows the user to switch different control methods; thus, the user can select the preferred control method. The users did not prefer the standing mode of gesture control using the Kinect sensor because of fatigue. However, the mode still could be used for rehabilitation purposes and occasionally forced by the doctor.

The new version of the robot – Robco 19, described in this paper, was successfully tested with real users and the new modern gesture control using the leap motion sensor, and finally, the proposed multi-channel (multi-modal) architecture was successfully implemented and evaluated.

For future work, we plan to research the following tasks: introducing a new function for bringing heavy and difficult to reach objects for manipulation, a functionality which would provide rehabilitation training, and other additional functions according to the results from the questionnaire. We believe that these additional features could greatly improve the robot system, which will become more useful providing new functionalities to make everyday life of the elderly and disabled citizens much easier.

Acknowledgements

The research was partially supported by the Bulgarian National Science Fund (grants No. DN 07/23-15.12.2016 and KP-06-M27/1-04.12.2018), by the Slovak Research and Development Agency (grants No. APVV SK-TW-2017-0005, APVV-15-0731 and APVV-15-0517) and by the Cultural and Educational Grant Agency financed by the Ministry of Education, Science, Research and Sport of the Slovak Republic (grant No. KEGA 009TUKE-4/2019).

REFERENCES

[1] Broekens, J.—Heerink, M.—Rosendal, H.: Assistive Social Robots in Elderly Care: A Review. Gerontechnology, Vol. 8, 2009, No. 2, pp. 94–103.

[2] Körtner, T.: Ethical Challenges in the Use of Social Service Robots for Elderly People. Zeitschrift für Gerontologie und Geriatrie, Vol. 49, 2016, No. 4, pp. 303–307, doi: 10.1007/s00391-016-1066-5.

[3] Hosseini, S.—Goher, K.: Personal Care Robots for Older Adults: An Overview. Asian Social Science, Vol. 13, 2017, No. 1, pp. 11–19, doi: 10.5539/ass.v13n1p11.

[4] Portugal, D.—Santos, L.—Alvito, P.—Dias, J.—Samaras, G.—Christodoulou, E.: SocialRobot: An Interactive Mobile Robot for Elderly Home Care. 2015 IEEE/SICE International Symposium on System Integration (SII 2015), Nagoya, IEEE, 2015, pp. 811–816, doi: 10.1109/SII.2015.7405084.

[5] Kachouie, R.—Sedighadeli, S.—Khosla, R.—Chu, M.-T.: Socially Assistive Robots in Elderly Care: A Mixed-Method Systematic Literature Review. International Journal of Human-Computer Interaction, Vol. 30, 2014, No. 5, pp. 369–393, doi: 10.1080/10447318.2013.873278.

[6] Zielinska, T.: History of Service Robots. In: Ceccarelli, M. (Ed.): Service Robots and Robotics: Design and Application. IGI Global, 2012, pp. 1–14, doi: 10.4018/978-1-4666-0291-5.ch001.

[7] IFR, International Federation of Robotics, 2019. Retrieved May 21st, 2019, from `http://www.ifr.org/`.

[8] Chivarov, N.—Chikurtev, D.—Rangelov, I.—Markov, E.—Gigov, A.—Shivarov, N.—Yovchev, K.—Miteva, L.: Usability Study of Tele-Controlled Service Robot for Increasing the Quality of Life of Elderly and Disabled – "ROBCO 17". In: Aspragathos, N., Koustoumpardis, P., Moulianitis, V. (Eds.): Advances in Service and Industrial Robotics (RAAD 2018). Springer, Cham, Mechanisms and Machine Science, Vol. 67, 2019, pp. 121–131, doi: 10.1007/978-3-030-00232-9_13.

[9] Kim, S.-C.—Lee, B.-K.—Kim, C.-Y.: Usability Evaluation of Communication Service Robot for the Elderly. Journal of Back and Musculoskeletal Rehabilitation, Vol. 32, 2019, No. 2, pp. 313–319, doi: 10.3233/BMR-169655.

[10] Zahariev, R.—Valchkova, N.—Angelov, G.—Paunski, Y.: Cognitive Service Mobile Robots for Help of Disabled People. Proceedings of the International Conference "Robotics and Mechatronics and Social Implementations" 2018. Complex Control Systems, Vol. 1, 2018, pp. 74–79.

[11] Mast, M.—Burmester, M.—Graf, B.—Weisshardt, F.—Arbeiter, G.—Španěl, M.—Materna, Z.—Smrž, P.—Kronreif, G.: Design of the Human-Robot Interaction for a Semi-Autonomous Service Robot to Assist Elderly People. In: Wichert, R., Klausing, H. (Eds.): Ambient Assisted Living. Springer, Cham, Advanced Technologies and Societal Change, 2015, pp. 15–29, doi: 10.1007/978-3-319-11866-6_2.

[12] Chivarov, N.—Chikurtev, D.—Markov, E.—Chivarov, S.—Kopacek, P.: Cost Oriented Tele-Controlled Service Robot for Increasing the Quality of Life of Elderly and Disabled – ROBCO 18. IFAC-PapersOnLine, Vol. 51, 2018, No. 30, pp. 192–197, doi: 10.1016/j.ifacol.2018.11.285.

[13] Cheng, Y.—Wang, G. Y.: Mobile Robot Navigation Based on Lidar. 2018 Chinese Control and Decision Conference (CCDC), Shenyang, IEEE, 2018, pp. 1243–1246, doi: 10.1109/CCDC.2018.8407319.

[14] Ocando, M. G.—Certad, N.—Alvarado, S.—Terrones, Á.: Autonomous 2D SLAM and 3D Mapping of an Environment Using a Single 2D LIDAR and ROS. 2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR), Curitiba, IEEE, 2017, pp. 1–6, doi: 10.1109/SBR-LARS-R.2017.8215333.

[15] Chikurtev, D.: Indoor Navigation for Service Mobile Robots Using Robot Operating System (ROS). Problems of Engineering Cybernetics and Robotics, Vol. 67, Sofia, 2016, pp. 61–70.

[16] Babič, M.—Hluchy, L.—Krammer, P.—Matovič, B.—Kumar, R.—Kovač, P.: New Method for Constructing a Visibility Graph-Network in 3D Space and a New Hybrid System of Modeling. Computing and Informatics, Vol. 36, 2017, No. 5, pp. 1107–1126, doi: 10.4149/cai_2017_5_1107.

[17] Gatesichapakorn, S.—Takamatsu, J.—Ruchanurucks, M.: ROS Based Autonomous Mobile Robot Navigation Using 2D LiDAR and RGB-D Camera. 2019 First International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP), Bangkok, Thailand, IEEE, 2019, pp. 151–154, doi: 10.1109/ICA-SYMP.2019.8645984.

[18] Yankov, K.: Inverse Kinematics for Educational Robot MOVER 4. Applied Researches in Technics, Technologies and Education (ARTTE) – Journal of the Faculty

of Technics and Technologies, Trakia University, Vol. 5, 2017, No. 3, pp. 212–224, doi: 10.15547/artte.2017.03.009.

[19] Koubâa, A. (Ed.).: Robot Operating System (ROS). The Complete Reference (Volume 1). Springer, Cham, Studies in Computational Intelligence, Vol. 625, 2016, doi: 10.1007/978-3-319-26054-9.

[20] Rao, H. S.—Desai, V. H.—Bhat, R.—Jayaprakash, S.—Sampangi, Y.: A Study and Implementation of Mapping and Speech Recognition Techniques for an Autonomous Mobile Robot Based on ROS. International Journal of Advanced Mechatronic Systems, Vol. 7, 2017, No. 5, pp. 303–310, doi: 10.1504/IJAMECHS.2017.095874.

[21] Hendrich, N.—Bistry, H.—Zhang, J.: Architecture and Software Design for a Service Robot in an Elderly-Care Scenario. Engineering, Vol. 1, 2015, No. 1, pp. 27–35, doi: 10.15302/J-ENG-2015007.

[22] Chivarov, S.—Chikurtev, D.—Yovchev, K.—Chivarov, N.: Multi-Channel Software Infrastructure for Remote Control of Service Robots. 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), Paris, France, IEEE, 2019, doi: 10.1109/CoDIT.2019.8820362.

[23] Cheng, H.—Yang, L.—Liu, Z.: Survey on 3D Hand Gesture Recognition. IEEE Transactions on Circuits and Systems for Video Technology, Vol. 26, 2016, No. 9, pp. 1659–1673, doi: 10.1109/TCSVT.2015.2469551.

[24] Kishore, P. V. V.—Kameswari, P. S.—Niharika, K.—Tanuja, M.—Bindu, M.—Kumar, A. D.—Kumar, E. K.—Kiran, M. T.: Spatial Joint Features for 3D Human Skeletal Action Recognition System Using Spatial Graph Kernels. International Journal of Engineering and Technology, Vol. 7, 2018, No. 1.1, pp. 489–493, doi: 10.14419/ijet.v7i1.1.10152.

[25] Lun, R.—Zhao, W.: A Survey of Applications and Human Motion Recognition with Microsoft Kinect. International Journal of Pattern Recognition and Artificial Intelligence, Vol. 29, 2015, No. 5, Art. No. 1555008, doi: 10.1142/S0218001415550083.

[26] Sun, Y.—Li, C.—Li, G.—Jiang, G.—Jiang, D.—Liu, H.—Zheng, Z.—Shu, W.: Gesture Recognition Based on Kinect and sEMG Signal Fusion. Mobile Networks and Applications, Vol. 23, 2018, No. 4, pp. 797–805, doi: 10.1007/s11036-018-1008-0.

[27] Cicirelli, G.—Attolico, C.—Guaragnella, C.—D'Orazio, T.: A Kinect-Based Gesture Recognition Approach for a Natural Human Robot Interface. International Journal of Advanced Robotic Systems, Vol. 12, 2015, No. 3, pp. 1–22, doi: 10.5772/59974.

[28] Lu, W.—Tong, Z.—Chu, J.: Dynamic Hand Gesture Recognition with Leap Motion Controller. IEEE Signal Processing Letters, Vol. 23, 2016, No. 9, pp. 1188–1192, doi: 10.1109/LSP.2016.2590470.

[29] McCartney, R.—Yuan, J.—Bischof, H.-P.: Gesture Recognition with the Leap Motion Controller. Accessed from `https://scholarworks.rit.edu/other/857`. 2015.

[30] Artal-Sevil, J. S.—Montañés, J. L.—Acón, A.—Domínguez, J. A.: Control of a Bionic Hand Using Real-Time Gesture Recognition Techniques Through

Leap Motion Controller. 2018 XIII Technologies Applied to Electronics Teaching Conference (TAEE), La Laguna, Spain, IEEE, 2018, pp. 1–7, doi: 10.1109/TAEE.2018.8476122.

[31] Chivarov, N.—Shivarov, N.: Remote Control User Interfaces for Service Mobile Robots for Elderly Care. IFAC-PapersOnLine, Vol. 49, 2016, No. 29, pp. 73–76, doi: 10.1016/j.ifacol.2016.11.104.

[32] Chikurtev, D.—Yovchev, K.—Chikurtev, E.: Design and Functionality of Web User Interface for Control of Service Mobile Robot Through the Internet. Problems of Engineering Cybernetics and Robotics, Vol. 67, 2016, pp. 51–60.

[33] FlexBE, 2019. Retrieved July 1st, 2019, from `http://wiki.ros.org/flexbe`.

[34] Buyval, A.—Afanasyev, I.—Magid, E.: Comparative Analysis of ROS-Based Monocular SLAM Methods for Indoor Navigation. Proceedings of the Ninth International Conference on Machine Vision (ICMV 2016), Proceedings of the SPIE, Vol. 10341, 2017, Art. No. 103411K, doi: 10.1117/12.2268809.

[35] Li, Y.—Shi, C.: Localization and Navigation for Indoor Mobile Robot Based on ROS. 2018 Chinese Automation Congress (CAC), Xi'an, China, IEEE, 2018, pp. 1135–1139, doi: 10.1109/CAC.2018.8623225.

[36] Chikurtev, D.—Rangelov, I.—Chivarov, N.—Shivarov, N.—Gigov, A.: Control of Service Robot via Voice Commands. Problems of Engineering Cybernetics and Robotics, Vol. 69, 2017, pp. 62–67.

[37] Carruth, D. W.—Hudson, C. R.—Bethel, C. L.—Pleva, M.—Ondas, S.—Juhar, J.: Using HMD for Immersive Training of Voice-Based Operation of Small Unmanned Ground Vehicles. In: Chen, J., Fragomeni, G. (Eds.): Virtual, Augmented and Mixed Reality. Applications and Case Studies (HCII 2019). Springer, Cham, Lecture Notes in Computer Science, Vol. 11575, 2019, pp. 34–46, doi: 10.1007/978-3-030-21565-1_3.

[38] Ondas, S.—Gurcik, M.: Domain-Specific Language Models Training Methodology for the In-Car Infotainment. Intelligent Decision Technologies, Vol. 11, 2017, No. 4, pp. 417–422, doi: 10.3233/IDT-170310.

[39] Machová, K.—Mikula, M.—Szabóová, M.—Mach, M.: Sentiment and Authority Analysis in Conversational Content. Computing and Informatics, Vol. 37, 2018, No. 3, pp. 737–758, doi: 10.4149/cai_2018_3_737.

[40] Liao, Y.-F.—Pleva, M.—Hladek, D.—Stas, J.—Viszlay, P.—Lojka, M.—Juhar, J.: Gated Module Neural Network for Multilingual Speech Recognition. 2018 11th International Symposium on Chinese Spoken Language Processing (ISCSLP), Taipei City, Taiwan, IEEE, 2018, pp. 131–135, doi: 10.1109/ISCSLP.2018.8706679.

[41] Hudson, C.—Bethel, C. L.—Carruth, D. W.—Pleva, M.—Juhar, J.—Ondas, S.: A Training Tool for Speech Driven Human-Robot Interaction Applications. 2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA 2017), Danvers, IEEE, 2017, pp. 167–172, doi: 10.1109/ICETA.2017.8102488.

[42] Ondáš, S.—Pleva, M.—Krištan, R.—Husovský, R.—Juhár, J.: VoMIS – The VoiceXML-Based Multimodal Interactive System for NAO Robot. 2018 IEEE World Symposium on Digital Intelligence for Systems and Machines (DISA 2018), IEEE, 2018, pp. 315–319, doi: 10.1109/DISA.2018.8490598.

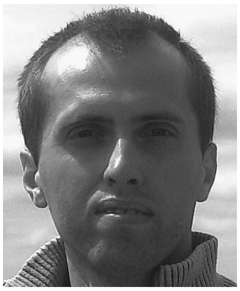
[43] Chivarov, N.—Chivarov, S.—Yovchev, K.—Chikurtev, D.—Shivarov, N.: Intelligent Modular Service Mobile Robot ROBCO 12 for Elderly and Disabled Persons Care. 2014 23rd International Conference on Robotics in Alpe-Adria-Danube Region (RAAD), Smolenice, Slovakia, 2014, pp. 343–348, doi: 10.1109/RAAD.2014.7002238.

Nayden Chivarov received his Ph.D. degree from the Institute of Handling Device and Robotics, TU Vienna, Austria. Currently he is head of RILab i2030 in the Institute of Information and Communication Technologies – BAS and Associated Professor at the EPU. His research interests are: service robots, mechatronics, human-robot interaction, Industry 4.0, IoT.

Denis Chikurtev received his Ph.D. degree from the Institute of Information and Communication Technologies at Bulgarian Academy of Sciences in 2017. He is currently Conduct Researcher in the field of robotics and ICT and running teaching courses in internet technologies. His major research interests are in the area of robotics: hardware and software architectures, computer vision and indoor localization, and navigation applicable for industrial robotic manipulators or mobile service robots.

Stefan Chivarov is currently working towards his Ph.D. in the Institute of Mechanics and Mechatronics – IHRT, TU Vienna, Austria. His research interests are: software architecture, autonomous navigation and localization, robotics, smart technologies, IoT.

Matus Pleva received his Ph.D. degree in telecommunications at the Department of Electronics and Multimedia Communications of the Faculty of Electrical Engineering and Informatics at the Technical University of Košice in 2010. He works as Assistant Professor in the field of acoustic modeling, acoustic event detection, speaker recognition, speech processing, human-machine interaction, security and biometrics, networking, etc. He also participated in more than 30 national and international projects and COST actions. He has published over 100 technical papers in journals and conference proceedings.



Stanislav Ondas graduated at the Department of Electronics and Multimedia Communications of the Faculty of Electrical Engineering and Informatics at the Technical University of Košice in 2004. He received his Ph.D. degree at the same department in the field of telecommunications in 2008. He is currently working as Assistant Professor in the Laboratory of Speech and Mobile Technologies at the same department. He is a specialist in the field of human-machine interaction, dialogue modelling and management, natural language processing and semantic analysis.

Jozef Juhar graduated from the Technical University of Košice in 1980. He received his Ph.D. degree in radioelectronics from the Technical University of Košice in 1991, where he works as Full Professor at the Department of Electronics and Multimedia Communications. He is author and co-author of more than 200 scientific papers. His research interests include digital speech and audio processing, speech/speaker identification, speech synthesis, development in spoken dialogue and speech recognition systems in telecommunication networks.

Kaloyan Yovchev received his Ph.D. degree from the Faculty of Mathematics and Informatics at Sofia University "St. Kliment Ohridski" in 2018. He is currently teaching courses in computer science. His major research interest is development and implementation of iterative learning control methods for industrial robotic manipulators. Algorithms for robust control of unmanned robotic systems are also in his research area.