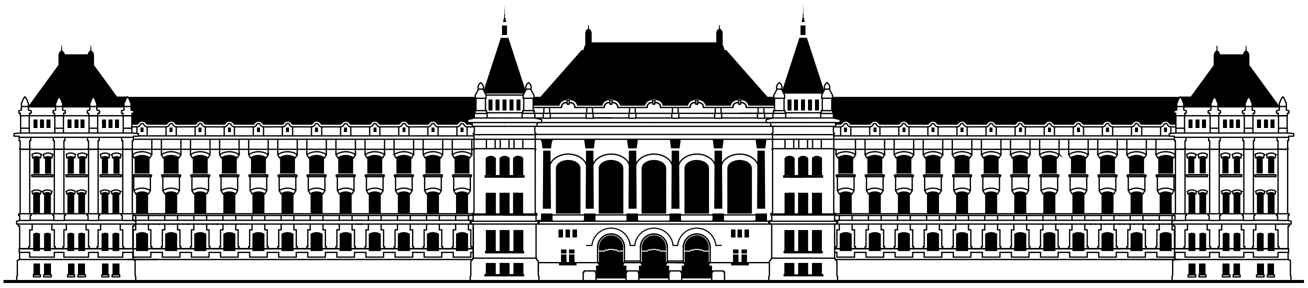
PROCEEDINGS OF THE 30TH MINISYMPOSIUM

OF THE

DEPARTMENT OF MEASUREMENT AND INFORMATION SYSTEMS
BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS

(MINISY@DMIS 2023)

FEBRUARY 6–7, 2023
BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS



BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS
DEPARTMENT OF MEASUREMENT AND INFORMATION SYSTEMS

ISBN 978-963-421-904-0

Homepage of the Conference:
`http://minisy.mit.bme.hu/`

Sponsored by:
Schnell László Foundation

[1]This list contains all the Reviewers who participated in the review process of papers published in this proceedings, as well as of research reports that were presented at the Minisymposium:
Balázs Bank, Bence Bolgár, András Földvári, László Gönczy, Gábor Hullám, Gábor Huszerl, Ákos Jobbágy, Attila Klenik, Imre Kocsis, Zsolt Kollár, István Majzik, Zoltán Micskei, Vince Molnár, Béla Pataki, Vilmos Pálfi, Oszkár Semeráth, Dóra Varnyú and András Vörös

Foreword

On behalf of the Organizing Committee, I welcome you to the 30th Minisymposium of the Department of Measurement and Information Systems at the Budapest University of Technology and Economics. As we celebrate this jubilee, it is an honor to announce that this is the first time we can organize the event in the frame of VIK Inference, the Faculty of Electrical Engineering and Informatics Conference. The significant number of presentations opens the way to having a full two-day event, again. This joyful result has been gradually achieved as we have observed an increasing number of Ph.D. students over the last couple of years. We hope this tendency will also remain unbroken in the upcoming years.
While regular presentations of our Ph.D. students give the backbone of the conference, the participants will have the opportunity to gain insight into the SMART4ALL European Union project. Besides, our talented IMSc students will present their results in a separate poster session.
The topics of the presentations express the diversity of the Department: we will have talks from the area of digital signal processing, embedded systems, artificial intelligence, and fault tolerant systems.
I hope that the two days of the 30th Minisymposium will expose the details of our research activities, as well as will imply fruitful conversations that can subserve to take the following step.

Budapest, February 6. 2023

[illegible]

Balázs Renczes
General Chair

Contents

Bence Graics and István Majzik:
Integration Test Generation and Formal Verification for Distributed Controllers 1
Kristóf Horváth, Balázs Bank and György Orosz:
Towards Automated Worst-Case Analysis of Circuits: Selecting Initial Values for Global Optimization 5
Dániel Szekeres and István Majzik:
Lazy Abstraction for Probabilistic Systems 9
András Palkó and László Sujbert:
Improving the Convergence Speed of the Resonator-Based Observer by Significant Component Selection 13
Bence Ország and László Sujbert:
Behavior of Observer Performing Discrete Gabor Transform 17
Mihály Vetró and Gábor Hullám:
Analyzing the Discriminative Power of EEG Microstates Over Mental Tasks 21
Gábor Révy and Dániel Hadházi:
Thoratic Spine Segmentation Based on CT images 25
András Földvári and András Pataricza:
Handling Uncertainty in Error Propagation Analysis 29
Bertalan Zoltán Péter and Imre Kocsis:
N-Version Programming as a Mitigation for Smart Contract Faults in Execute-Order-Validate Blockchain Systems 33
Noor Al-Gburi and Imre Kocsis:
Knowledge-driven Exploratory Performance Data Analysis for Execute-Order-Validate Blockchains 37
Dániel Sándor and Péter Antal:
Transfer Learning in Heterogeneous Drug-Target Interaction Predictions Using Federated Boosting 41

Integration Test Generation and Formal Verification for Distributed Controllers

Bence Graics, István Majzik
Budapest University of Technology and Economics,
Department of Measurement and Information Systems
Budapest, Hungary
Email: {graics, majzik}@mit.bme.hu

***Abstract*—Software-intensive distributed controllers are becoming increasingly prevalent, among others, also in railway interlocking systems (RIS). As such systems carry out critical tasks, their systematic verification and testing are a must, which can be supported by formal methods. This paper presents a verification and testing approach for a distributed RIS subsystem using hidden formal methods. The subsystem's functional behavior is modeled using statechart components defined in a high-level UML-based modeling language, which are integrated according to sound execution and interaction semantics defined by the RIS protocol. The emergent model is automatically mapped into input formalisms of model checker back-ends. Integration tests for the system implementation are derived according to various model-based coverage criteria using the model checker back-ends and generated properties. The approach is implemented in our open source Gamma Statechart Composition Framework.**

***Index Terms*—MBSE, collaborating statecharts, hidden formal methods, model checking, test generation, integrated tool suite**

I. Introduction

Software-intensive programmable controllers are getting more and more prevalent in critical infrastructure, e.g., in railway interlocking systems (RIS). Such systems are generally embedded into their dynamic environment and must coordinate multiple subsystems/components to carry out complex tasks in response to external commands or environmental changes.

The distributed nature of these systems encumbers their design, necessitating precise means to describe the integration of system components, including their execution and communication, in addition to their standalone behavior. Moreover, as these systems carry out critical tasks, the automated formal verification of their design and the testing of their implementation are a must in the development process.

Model-based and component-based systems engineering (MBSE and CBSE) [1], [2] approaches promote the use of reusable models and components based on high-level modeling languages. However, they usually do not provide sophisticated tool-centric means for the automated verification of the design artifacts or the testing of the system implementation, either due to informal model descriptions or the lack of sound and efficient verification methods.

These issues are also prevalent in the railway domain: [3] and [4] argue that the main barriers that hinder the wide adoption of verification-oriented MBSE/CBSE approaches stem from the lack of traceability and process integration between the high-level models and verification back-ends.

This paper offers solutions to these shortcomings and presents a verification and test generation approach (based on hidden formal methods) for distributed controllers, adapted to the design language and integration semantics of a specific RIS design. The approach relies on the proprietary EXtended State Machine Language (XSML) used in a railway project to model the functional behavior of system components. The models are transformed into the statechart language (GSL) [5] of our Gamma Statechart Composition Framework [6], a framework for the component-based design and verification of reactive systems. The models are integrated in Gamma's composition language (GCL) [7] based on sound execution and communication semantics in accordance with the system specification. The composite model is then mapped into input formalisms of model checker back-ends to support the exhaustive verification of its behavior and generate integration tests for the implementation based on various coverage criteria. The mappings feature model reduction and slicing algorithms to support industrial-scale systems.

Similar frameworks have been introduced in [8], [9] and [10] for the verification of and test generation for component-based reactive systems. However, these approaches rely on commercial tools, hindering their extensibility, and do not provide flexibility in terms of integration semantics, contrary to our approach building on the open source Gamma framework.

Our novel contributions are *(1)* the transformation of XSML models into GSL, focusing on the languages' characteristics, *(2)* a new composition mode in GCL for the semantic-preserving integration of XSML components, and *(3)* the application of our approach on a real-life distributed RIS subsystem under development.

II. Mapping XSML Components Into GSL

XSML is a textual statechart language reusing most elements of UML, e.g., it offers hierarchical states and transitions for describing state-based behavior and variables for expressing memory. However, as it is designed to describe critical functionalities, it aims for the easy interpretability of the models and discards UML elements for complex transitions, e.g., choice, merge, fork and join nodes, history states, as well

This work was supported by the ÚNKP-22-3 New National Excellence Program of the Ministry for Innovation and Technology.

as entry and exit actions of states. Instead, it offers a powerful action language to capture the handling of variables as well as control flow (target states) in transitions. It also refines UML's operational semantics to *eliminate nondeterminism* in regard to transitions and orthogonal regions by introducing *priorities* and thus, a *sequential execution* of these elements.

Listing 1 presents an excerpt of one of the RIS components, called *ObjectHandler*, defined in XSML. The excerpt describes a state where the component waits for the confirmation message of a certain request. The waiting can end in an expected confirmation (entering *CmdConfirm*) or a fallback to a previous state (*WaitTS1Req*) via transitions due to an invalid message or a timeout while managing variables (potentially through functions) and dispatching messages via ports.

```
// Integer variables with different bitlengths
U8 SessionID;
U32 TimestampObj1, TimestampVk1;
...
state WaitTS2Req {
 // Timeout based on a parameter value
 timeout after (TimeConfirmationTimeout) {
  change WaitTS1Req;
 }
 // Transition triggered by a Rigel message
 event Rigel msg [msg.MsgType == MsgType.RigelMsgReqTs2]
     from PortIn {
  // Effects described in an action language
  if (SessionID != msg.SessionId)
   change CmdConfirm; // Potential target state
  else {
   var Rigel ansTs1Msg = ProcessReqTs2( // Function call
     TimestampVk1, TimestampObj1, SessionID);
   send ansTs1Msg to PortOut; // Message dispatch
   change WaitTS1Req; // Potential target state
  }
 }
}
```

Listing 1: Excerpt from the *ObjectHandler* XSML model.

The XSML-GSL model transformation must handle two characteristics of XSML besides the straightforward mapping of most model elements, e.g., regions, states and variables:

1) *Selecting target states* in transitions: In XSML, a single syntactic structure (if-else branches) is used to process an event, specifying not only the actions but also different target states (*change* statements) depending on the conditional branches; in contrast to GSL, where each transition shall have only a single, fixed target state.
2) *Introducing priorities*: In XSML, priorities of transitions and orthogonal regions are used to ensure deterministic behavior, which have to be represented in GSL.

Listing 2 illustrates how the transformation handles *target state selection* during event processing. First, a *transition* triggered by the corresponding event is introduced, which enters a *choice state*. This transition executes the corresponding actions while setting *auxiliary boolean variables* (*toWaitTS1Req* and *toCmdConfirm*) that identify the corresponding target states. From the choice state, a *set of transitions* is used, where each transition enters a proper target state depending on the values of the auxiliary variables referenced from their guard expressions. Note that this mapping retains the *atomicity* of transitions due to the semantics of choice states in GSL [5].

Introducing priorities to transitions and orthogonal regions is based on so-called *semantic variation points* offered by GSL, which – among others – support adjusting

- the *execution* of actions in orthogonal regions of composite states, which can be *sequential*, i.e., in the order of the declaration of regions, *unordered*, i.e, any region permutation is considered valid, and *parallel*, i.e., actions in orthogonal regions can interleave in any way; and,
- *priority* between enabled transitions leaving the same state – the *absence* of priority leads to *nondeterministic* choices between enabled transitions during execution.

With respect to the operational semantics of XSML, the transformation applies the *sequential* execution of orthogonal regions and transitions *prioritized* according to their *order of definition* ("earlier" defined transitions have a higher priority).

```
// Auxiliary boolean variables for target state selection
var toWaitTS1Req, toCmdConfirm : boolean
..
// Choice state for target state selection
choice WaitTS2Req_
..
// Selecting target states: single transition
transition from WaitTS2Req to WaitTS2Req_ when
     PortIn.msg [PortIn.msg::Value.MsgType ==
     MsgType::RigelMsgReqTs2] / {
 if (SessionID != PortIn.msg::Value.SessionId)
  toCmdConfirm := true; // Setting target state
 else {
  var ansTs1Msg : Rigel := ProcessReqTs2( // Function call
     TimestampVk1, TimestampObj1, SessionID);
  raise PortOut.message(ansTs1Msg);
  toWaitTS1Req := true; // Setting target state
 }
}
// Selecting target states: set of transitions
transition from WaitTS2Req_ to WaitTS1Req [toWaitTS1Req]
transition from WaitTS2Req_ to CmdConfirm [toCmdConfirm]
```

Listing 2: GSL elements derived from the transition triggered by a Rigel message in Listing 1.

III. Integrating XSML Components

Similarly to standalone statecharts, XSML also aims for a *deterministic* behavior at the level of component integration. Accordingly, it defines deterministic execution and communication semantics for integrated (composite) components that feature *(1)* the *sequential execution* of contained components and *(2)* their communication using immutable *messages* stored in *prioritized message queues*. Consequently, GCL must provide a composition mode that conforms to these characteristics and thus, we introduce the new *scheduled asynchronous-reactive* composition mode as previously introduced composition modes feature *parallel* execution with *message*-based communication (*asynchronous-reactive*) or *signal*-based communication (*cascade* and *synchronous-reactive*) [7].

The examined RIS subsystem represents the realization of the so-called Rigel protocol and comprises three components, namely *controlCenter*, *dispatcher* and *objectHandler*. Listing 3 describes the RIS subsystem model integrated in GCL using the *scheduled asynchronous-reactive* composition mode. The model has an integer parameter (*Timeout*) and two ports (*ControlPortIn*, *ControlPortOut*) for the transmission of input and output messages defined in the *Rigel* interface. The model contains the above-mentioned standalone statechart components (the *objectHandler* also has a *Timeout* parameter) derived from XSML models whose ports are connected using *channels* to enable internal communication. Moreover, the control ports of the *controlCenter* component are *bound* to the external ports of the system.

The new composition mode supports a *cycle-based* execution mode in which components are executed *sequentially*. The execution order is defined in an execution list (*execute* keyword). A component can be referenced multiple times in the execution list, allowing its multiple execution in a single cycle. Regarding communication, the components interact using immutable *messages* stored in prioritized *message queues*. Such message queues can be defined using *asynchronous adapter* [7] models (*ControlCenter*, *Dispatcher* and *ObjectHandler*) that adapt statechart models to message-based communication. A message queue has the following fixed attributes: *(1)* stored message types, *(2)* priority, *(3)* capacity and *(4)* the handling of incoming messages in case the queue is full (discard the *incoming* or the *oldest* stored message). When selecting a message for processing, one is always retrieved from the highest priority non-empty queue.

```
scheduled-async RIS(Timeout : integer) [
 // System ports visible from the environment
 port ControlPortIn : requires Rigel
 port ControlPortOut : provides Rigel
] {
 // Contained components of the RIS
 component controlCenter : ControlCenter
 component dispatcher : Dispatcher
 component objectHandler : ObjectHandler(Timeout)
 // Binding the control center ports to the system ports
 bind ControlPortIn -> controlCenter.ControlPortIn
 bind ControlPortOut -> controlCenter.ControlPortOut
 // Channels for the inter-component communication
 channel [ controlCenter.PortOut ] -o)- [ dispatcher.
    ControlCenterPortIn ]
 channel [ dispatcher.ControlCenterPortOut ] -o)- [
    controlCenter.PortIn ]
 channel [ dispatcher.ObjectHandlerPortOut ] -o)- [
    objectHandler.DispatcherPortIn ]
 channel [ objectHandler.DispatcherPortOut ] -o)- [
    dispatcher.ObjectHandlerPortIn ]
 // Scheduling order of components
 execute controlCenter, dispatcher, objectHandler
}
```

Listing 3: RIS model integrated in GCL using the *scheduled asynchronous-reactive* composition mode.

IV. Formal Verification

The complete GCL model is mapped into low-level analysis models via a sequence of internal automated model transformations that take into account the composition mode. The analysis models can be verified with respect to manually defined properties using model checker back-ends integrated to Gamma. The results, i.e., whether the property holds in the model and potentially a diagnostic trace as proof, are automatically back-annotated to the source GCL model. Currently, UPPAAL, Theta and Spin are supported as back-ends, which are tailored to handling different models, e.g., UPPAAL supports timed behavior, Theta supports abstraction-based symbolic techniques, and Spin excels at checking parallel behavior. They also support different property specification languages, e.g., UPPAAL supports a restricted CTL, Theta supports reachability, and Spin supports LTL [11], providing a good portfolio for model checking.

In order to support the exhaustive verification of industrial-scale systems, the transformations feature several *model reduction* and *slicing* algorithms to reduce the state space of models under verification. The model reduction algorithms, which are independent of the verifiable properties and applied on the model in itself, reduce the following model elements:

- unused variables and input events with their parameters;
- unfireable transitions, e.g., due to the lack of triggering events or guards evaluating to constant false;
- unreachable states and regions without a functionality, e.g., with a single simple state without entry/exit actions.

Model slicing is conducted depending on the verifiable properties and reduce the following model elements:

- unreferenced enumeration literals;
- unreferenced variables and input events with their parameters that do not influence internal behavior.

V. Test Generation

Test generation based on the complete GCL model utilizes the verification functionalities presented in Sect. IV. As a general idea, in a testing context, a diagnostic trace for a GCL model derived during formal verification can be considered as an *abstract test case* for the property based on which it is generated, representing a test target. Thus, with the goal of generating tests, we control model checkers in a way that they generate diagnostic traces (abstract test cases) to cover test targets specified as formal properties (*trap properties*) [12]. These abstract test cases then can be customized to different execution environments, e.g., Java and C.

Test targets can be specified based on the following model element based (structural), behavior- (interactional) and dataflow-based coverage criteria:

- *output event*, *state*, *transition* and *transition-pair* (pairs of transitions entering and leaving a certain state);
- *sending* (event raise) and *receiving/processing* (transition triggered by the event) of an event between two *communicating components*;
- *execution paths* between the definition (def) and the use/reading (use) of variables within standalone components and also between communicating components.

The generated tests can be used to detect faults in component implementations (e.g., missing implementation of transitions), interaction of components, and improper variable definitions and uses in system implementations.

Test targets are defined in terms of reachability properties, which are trivial only in the case of output events and states as these model elements can be directly referenced from properties. In other cases, the GCL model has to be *annotated* to enable describing the coverage of these criteria. Accordingly, transition and variable def-use coverage criteria necessitate the injection of boolean variables indicating their coverage, whereas transition-pair and interaction coverage criteria require the injection of integer variables that store the ID of the covered elements. As the number of coverable elements can be large, the approach supports the *customization* of criteria, allowing the inclusion/exclusion of components and relevant model elements, e.g., states, transitions and ports.

In order to make test generation more efficient in terms of time and the size of the generated test set, the approach utilizes two optimization algorithms. After generating a *new abstract test case*, the first algorithm iterates through the still *uncovered* test targets and checks whether the test case also covers some of them [12]; such test targets get discarded. After covering *each test target*, the second algorithm is applied, which searches for test cases in the test set that are *prefixes* of other test cases [13]. Note that such test cases can exist even when the first algorithm is applied due to the nondeterministic order of processing test targets. Such test cases do not contribute to the coverage of additional criteria, and thus, can be discarded to further reduce the generated test set.

VI. Evaluation

We evaluated the feasibility of our test generation approach on the integrated RIS model, focusing on test generation *time* and the *size* of the generated test set *with test optimization* in the case of full *state*, *transition* and *interaction* coverage. The RIS model altogether has 22 regions, 38 states, 118 transitions, 10 variables and 13 clock variables. We set a 5ms timeout parameter value for the model and used UPPAAL as this back-end could manage the features of the RIS the most efficiently in terms of execution time. Table I contains the measurement results. We generated tests five times for each coverage criterion; the time-related values in the table are represented in seconds and refer to the median of these results (the test size related values do not change in different runs).

The results show that as the coverage criterion for testing gets finer, the number of test targets, generated tests and contained cycles increases; apart from one case concerning interaction coverage due to the large number of uncoverable interactions. In addition, the average generation time for a single test target also increases due to the complexity of injected annotations (auxiliary variables). Nevertheless, the results show the approach is feasible for industrial-scale models, as *every test target* for every criterion could be handled in less than 14 seconds on average without any complication, e.g., a timeout or out of memory error in the process.

	State	**Transition**	**Interaction**
#Test targets	38	118	387
#Generated tests	4	26	22
#Cycles in tests	30	230	240
$\Sigma\mathbf{T}$ (s)	243	950	5377
$\overline{\mathrm{T}}$ (s)	6.4	8.1	13.9

TABLE I: The number of *test targets*, *generated tests* and *cycles* in the generated tests, as well as the median joint *test generation time* and *average test generation* time for a *single test target* in *seconds* for full *state*, *transition* and *interaction* coverage in the integrated RIS model.

VII. Conclusion and Future Work

In this paper, we presented a formal verification and model-based integration test generation approach for distributed controllers, adapted to a real-life RIS subsystem. The adaptation necessitated the mapping of the XSML design language and its semantics into the internal languages of the Gamma framework and also the introduction of a new composition mode. Based on these extensions, automated formal verification and customizable test generation are supported for RIS design models. Our evaluation demonstrated the feasibility of the approach on an existing distributed RIS subsystem using different coverage criteria for test generation.

Subject to future work, we plan to extend the approach to support additional execution and communication modes during component integration, e.g., introduce shared *global message queues* for components, to aid engineers in experimenting with different integration semantics.

References

[1] A. Childs, J. Greenwald, G. Jung, M. Hoosier, and J. Hatcliff, “Calm and Cadena: Metamodeling for component-based product-line development,” *Computer*, vol. 39, no. 2, pp. 42–50, 2006.

[2] J. Sztipanovits, T. Bapty, S. Neema, L. Howard, and E. Jackson, *OpenMETA: A Model- and Component-Based Design Tool Chain for Cyber-Physical Systems*. Berlin, H.: Springer, 2014, pp. 235–248.

[3] A. Ferrari, F. Mazzanti, D. Basile, and M. H. ter Beek, “Systematic evaluation and usability analysis of formal methods tools for railway signaling system design,” *IEEE Transactions on Software Engineering*, vol. 48, no. 11, pp. 4675–4691, 2022.

[4] G. Lukács and T. Bartha, “Formal modeling and verification of the functionality of electronic urban railway control systems through a case study,” *Urban Rail Transit*, vol. 8, 11 2022.

[5] B. Graics, “Documentation of the Gamma Statechart Composition Framework v0.9,” Budapest Univ. of Technology and Economics, Tech. Rep., 2016, https://tinyurl.com/yeywrkd6.

[6] V. Molnár, B. Graics, A. Vörös, I. Majzik, and D. Varró, “The Gamma Statechart Composition Framework,” in *40th International Conference on Software Engineering (ICSE)*. Gothenburg, Sweden: ACM, 2018, p. 113–116.

[7] B. Graics, V. Molnár, A. Vörös, I. Majzik, and D. Varró, “Mixed-semantics composition of statecharts for the component-based design of reactive systems,” *Software and Systems Modeling*, vol. 19, p. 1483–1517, 2020.

[8] S. Mohalik, A. A. Gadkari, A. Yeolekar, K. Shashidhar, and S. Ramesh, “Automatic test case generation from Simulink/Stateflow models using model checking,” *Softw. Test. Verif. Reliab.*, vol. 24, pp. 155–180, 2014.

[9] A. Hartman and K. Nagin, “The AGEDIS tools for model based testing,” *ACM Sigsoft Software Engineering Notes*, vol. 29, 07 2004.

[10] G. Hamon, L. de Moura, and J. Rushby, “Generating efficient test sets with a model checker,” in *Proceedings of the Second International Conference on Software Engineering and Formal Methods (SEFM)*, 01 2004, pp. 261–270.

[11] E. A. Emerson and J. Y. Halpern, ““Sometimes” and “not never” revisited: On branching versus linear time temporal logic,” *J. ACM*, vol. 33, no. 1, p. 151–178, Jan. 1986.

[12] G. Fraser, F. Wotawa, and P. E. Ammann, “Testing with model checkers: a survey,” *Software Testing, Verification and Reliability*, vol. 19, no. 3, pp. 215–261, 2009.

[13] R. Dorofeeva, K. El-Fakih, S. Maag, A. Cavalli, and N. Yevtushenko, “Experimental evaluation of FSM-based testing methods,” in *Third IEEE International Conference on Software Engineering and Formal Methods (SEFM’05)*, 2005, pp. 23–32.

Towards Automated Worst-Case Analysis of Circuits: Selecting Initial Values for Global Optimization

Kristóf Horváth, Balázs Bank, György Orosz
Budapest University of Technology and Economics
Department of Measurement and Information Systems
Budapest, Hungary
Email: {hkristof, bank, orosz}@mit.bme.hu

***Abstract*—Worst-case circuit analysis is a mandatory practice in hardware verification and validation. To this end, several methods, including extreme value analysis (EVA) and Monte Carlo analysis are commonly used, however, each has its own limitations. Numerical optimization-based methods have the potential to be generally usable, but have the tendency to get stuck in a local minimum, which can be mitigated using carefully chosen initial values. In this paper we propose methods for automated initial value selection for black-box circuit models. The methods are demonstrated to work on several standard test functions, which is a first step in building an automated worst-case circuit analysis tool.**

***Index Terms*—worst-case circuit analysis, numerical optimization, initial values**

I. Introduction

Worst-case analysis (WCA) is an important step in hardware design verification [1]. Because the parameters of the circuit components can vary in production, certain circuit properties (e.g. amplifier gain, attenuation, power dissipation, etc.) should be analysed to make sure that parameter variation will not push the circuit outside its limits.

For this purpose, various methods have been developed over the years. Among them, extreme value analysis (EVA) is a popular tool in the industry. EVA is based on the assumption that the extreme values of circuit properties can be found on the boundaries of parameter values, therefore by evaluating all extreme-value combinations it is possible to find those which result in the highest deviation from nominal circuit properties. The drawback of EVA is that it requires 2^N function evaluation where N is the number of parameters. In addition, for some circuits the assumption of having the extreme value at the boundaries is not valid (see Figure 2), and thus EVA leads to incorrect results.

Monte-Carlo-analysis is another traditional method, where simulations are used to calculate a probability distribution estimator for the circuit property in question. It assumes that the circuit property is an approximately linear function of the component values, which might be a good approximation for many, but not for all parameters. The downsides of this method include a large computational demand as well as the fact that the results are not exact [2].

A more general approach to worst-case analysis is to perform it as an optimization task [3]. After all, the goal to WCA is to find the extreme values of a circuit property, which can be represented as a mathematical function. Tolerances of the component parameters can be incorporated into the optimization problem as boundaries to the search space.

Traditional optimization methods (e.g. gradient descent, direct methods, etc.) assume cost functions that have only one local minimum in the search space. These algorithms operate from a starting point, and make incremental steps that converge to a local optimum. In some circuit analysis problems, however, the function describing the circuit property in question can have multiple local optima, thus global optimization strategies are necessary. One approach is to sample the function and use clustering to select candidate points to be used as initial values and start a traditional local optimization algorithm from all of them [4], [5]. This way there will be at least one point from where the local optimizer can converge to the global optimum.

An additional difficulty in circuit analysis is that functions rarely appear in closed-form, rather than as a result of a Spice-based circuit simulation. Therefore, we only consider black-box functions for optimization targets, whose derivatives can not be evaluated.

In this paper we present methods for searching appropriate initial values for global optimization of black-box functions.

II. Example circuit

Consider the circuit in Figure 1. The objective of the analysis is the maximum power dissipation of transistor Q2. The tolerances for resistors are 5% of their nominal value, the transistors are used at temperatures between 0°C and 40°C and other parameters and tolerances came from the datasheet. The supply voltage, denoted by VCC, can be between 6 and 12 Volts, and the input voltage, marked by Uin, is allowed to be between 0 and 5.5 Volts. The number of parameters in this example is 14.

Using physical considerations, we have deduced that the power dissipation of Q2 is largely dependent on the input voltage of the circuit. The relationship is plotted in Figure 2 with all other parameters being fixed at the results of a preliminary worst-case analysis. It should be noted that there are two (local) maxima in this graph; at around 4 V and around 1 V, therefore global optimization is necessary.

DOI 10.3311/minisy2023-002

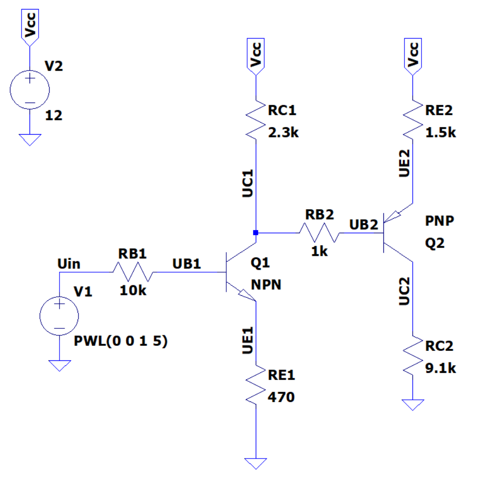


Fig. 1. A simple transistor-based circuit example. The property in question is the power dissipation of transistor Q2.

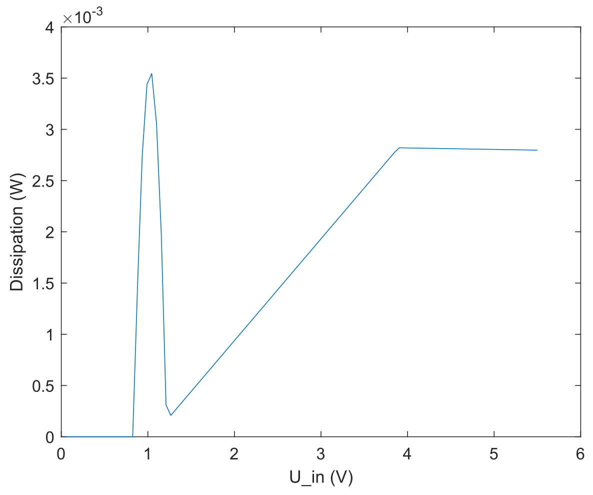


Fig. 2. Dissipation of the Q2 transistor in the example. Here, we only plotted the dissipation in terms of the input voltage, with the other parameters being fixed.

III. Methods for Determining a Set of Initial Values

Usually, initial values are chosen based on nominal values [3], or results of a pen-and-paper-based analysis. Another approach is to start the optimization from a random initial value. When the optimization is performed from several random initial values, it is likely that the optimization algorithm will converge to the global optimum from at least one set of initial values. The downside of using a large number of initial values is the excessive amount of redundant computations.

Our approach of determining a set of initial values is based on function sampling. The core idea has been used for empirical estimation of function shape and properties [6], [7] as well as by global optimization methods [4], [5]. The latter sample the function in a large number of random points and select only a few of them. Ideally, only one point is selected from the region of attraction around each local minimum, because selecting more than one will lead to unnecessary calculations in the subsequent local optimization step.

Obviously, an important condition for this approach is the proper sampling of the function of interest: for example, if the sample point set does not contain any points from the vicinity of a certain local minimum, then sampling-based methods can not find that local minimum.

Algorithm 1 Barrier search (BS) algorithm

```
1:  Generate random sampling points: ${\mathbf{x}_i}$ for $i = 1 \dots M$
2:  Evaluate function in sampling points: $y_i = f(\mathbf{x}_i)$ for $\forall \mathbf{x}_i$
3:  Make an ordered list: $f(\mathbf{x}_{o_1}) \leq f(\mathbf{x}_{o_2}) \leq \dots \leq f(x_{o_M})$
4:  Add first element to output set: $D = \{\mathbf{x}_{o_1}\}$ where $\mathbf{x}_{o_1} = \arg\min_{\mathbf{x}_i}(f(\mathbf{x}_i))$
5:  for all $\mathbf{x}_{o_k}$, $k = 2 \dots M$ do
6:      for all $\mathbf{x}_{d_i} \in D$ do
7:          Define midpoint as: $\mathbf{m}_{i,k} = (\mathbf{x}_{d_i} + \mathbf{x}_{o_k})/2$
8:          if $f(\mathbf{x}_{d_i}) < f(\mathbf{m}_{i,k})$ and $f(\mathbf{x}_{o_k}) < f(\mathbf{m}_{i,k})$ then
9:              $D := D \cup \{\mathbf{x}_{o_k}\}$
10:         else if $f(\mathbf{m}_{i,k}) < f(\mathbf{x}_{d_i})$ then
11:             Replace $\mathbf{x}_{d_i}$ in $D$ with $\mathbf{m}_{i,k}$
12:             Restart inner loop
13:         end if
14:     end for
15: end for
```

IV. Proposed Algorithms for Initial Value Selection

Our intention in constructing our algorithms was to keep them simple, yet at the same time use the least amount of function evaluations at searching suitable initial values for global optimization.

As a first step, all algorithms generate an arbitrary number of random sampling points, $\mathbf{x}_i$, and evaluate the objective function at these points: $y_i = f(\mathbf{x}_i)$. The output of each algorithm is a set of initial values, which we denote by $D = \{\mathbf{x}_{d_1}, \mathbf{x}_{d_2}, \dots, \mathbf{x}_{d_K}\}$, where $d_{1 \dots K}$ denote the indices selected as initial values from set $\{\mathbf{x}_i\}$.

Barrier search algorithm (BS) is based on the fact that two local minima should be separated by a barrier, i.e. there should be an area between two local minima where the function value is higher than any of the two minima. In practice, the condition is checked only at the midpoint between two testpoints. The pseudocode for BS algorithm can be found in Algorithm 1.

Convex definition check (CDC) algorithm is based on the necessary condition for convexity: in convex areas, any line segments connecting two points on the graph of the function lies above the graph between the two points. In practice, this condition is checked only at the midpoint between two testpoints. The pseudocode for CDC algorithm can be found in Algorithm 2.

Both the BS and CDC algorithms perform a simple preliminary optimization too: if the function value at the midpoint

Algorithm 2 Convex definition check (CDC) algorithm

```
1: Generate random sampling points: $\{\mathbf{x}_i\}$ for $i = 1 \dots M$
2: Evaluate function in sampling points: $y_i = f(\mathbf{x}_i)$ for $\forall \mathbf{x}_i$
3: Make an ordered list: $f(\mathbf{x}_{o_1}) \leq f(\mathbf{x}_{o_2}) \leq \dots \leq f(x_{o_M})$
4: Add first element to output set: $D = \{\mathbf{x}_{o_1}\}$ where $\mathbf{x}_{o_1} = \arg\min_{\mathbf{x}_i}(f(\mathbf{x}_i))$
5: for all $\mathbf{x}_{o_k}$, $k = 2 \dots M$ do
6:     for all $\mathbf{x}_{d_i} \in D$ do
7:         Define midpoint as: $\mathbf{m}_{i,k} = (\mathbf{x}_{d_i} + \mathbf{x}_{o_k})/2$
8:         if $f(\mathbf{m}_{i,k}) > (f(\mathbf{x}_{d_i}) + f(\mathbf{x}_{o_k}))/2$ then
9:             $D := D \cup \{\mathbf{x}_{o_k}\}$
10:        else if $f(\mathbf{m}_{i,k}) < f(\mathbf{x}_{d_i})$ then
11:            Replace $\mathbf{x}_{d_i}$ in $D$ with $\mathbf{m}_{i,k}$
12:            Restart inner loop
13:        end if
14:    end for
15: end for
```

Algorithm 3 Local minimum definition (LMD) algorithm

```
1: Generate random sampling points: $\{\mathbf{x}_i\}$ for $i = 1 \dots M$
2: Evaluate function in sampling points: $y_i = f(\mathbf{x}_i)$ for $\forall \mathbf{x}_i$
3: Make an ordered list: $f(\mathbf{x}_{o_1}) \leq f(\mathbf{x}_{o_2}) \leq \dots \leq f(x_{o_M})$
4: Add first element to output set: $D = \{\mathbf{x}_{o_1}\}$ where $\mathbf{x}_{o_1} = \arg\min_{\mathbf{x}_i}(f(\mathbf{x}_i))$
5: for all $\mathbf{x}_{o_k}$, $k = 2 \dots M$ do
6:     Find closest L points: $\|\mathbf{x}_{o_k} - \mathbf{x}_{c_1}\|_2 \leq \|\mathbf{x}_{o_k} - \mathbf{x}_{c_2}\|_2 \leq \dots \leq \|\mathbf{x}_{o_k} - \mathbf{x}_{c_L}\|_2$
7:     if $f(\mathbf{x}_{o_k}) < f(\mathbf{x}_{c_i})$ for $\forall i = 1 \dots L$ then
8:         $D := D \cup \{\mathbf{x}_{o_k}\}$
9:     end if
10: end for
```

($m_{i,k}$) is lower than the function value at the point in the output set (x_{d_i}), then the latter is replaced by the midpoint.

The third algorithm (Local minimum definition, LMD) is inspired by the definition of local minimum. For each sample point, the closest N points are considered: if the tested point has the lowest function value out of the closest L sampling points then it is assumed to be a local minimum. The pseudocode for LMD can be found in Algorithm 3. Note that LMD does not need additional function evaluations besides the initial sampling points.

V. Performance evaluation

In order to compare the proposed initial value search algorithms, we have tested them on some standard 2-dimensional test functions [8]: sphere, inverse sphere, Rosenbrock, Styblinski-Tang, Goldstein-Price, Booth, Matyas, Himmelblau, Three-hump camel, and McCormick functions. The reason for choosing these functions is that their properties are well-known, and the functions themselves are well-behaved and closely resemble typical physical systems that are analysed in WCA problems.

In our tests, we have sampled the functions in 1000 random points and used these samples for all tested algorithms. We have found that considering the closest 10 points in LMD algorithm leads to the lowest amount of redundant initial values. To measure performance, the number of initial values and the total number of function evaluations were used. The tests were performed on 30 different random point sets and the numbers of initial values returned by each algorithm were averaged. For finding the closest local optimum from each initial point, we have used Matlab's `fmincon` interior-point method as local optimizer.

Figure 3 shows the ratio between the average number of initial values returned by the algorithms and the number of real local minima of the functions. Additionally, the performances on the example circuit in Figure 1 are also shown. In all cases, the local optimization converged to either of the local minima, so a ratio of 1 means that the algorithm managed to find the region of attraction around each local minimum. Values higher than 1 mean that the algorithm returned more initial values than necessary, which lead to unnecessary calculations in the subsequent local optimization step. The BS algorithm could not find all of the local minima of Goldstein-Price function. This is caused by the fact that the function is relatively flat around its local minima and so there is a significant chance that the algorithm can not find a barrier between sampling points.

A box plot containing the required number of function evaluations can be found in Figure 4. It can be seen that the CD algorithm that is based on the definition of convexity requires more function evaluations than the other two in more than 50% of the cases. This is caused by the large number of initial points, as it returns all sampling points in a concave area of the function.

Probably the most surprising is that the LMD algorithm requires the least amount of function evaluations. This is because despite returning the highest number of initial points in the majority of the cases, it only needs to evaluate the function at the sampling points, which in our case is 1000 points. In later experiments we have found that this advantage diminishes as the number of dimensions increases.

VI. Multidimensional example

We have tested our algorithms on a scalable multidimensional function too:

$$f(\mathbf{x}) = -\sum_{i=1}^{N} \cos(2\pi x_i), \quad \forall x_i \in [0;1], \tag{1}$$

TABLE I
Ratio of missed regions of attraction and the ratio of duplicate points in regions of attraction for the multidimensional function example.

	Missed regions of attraction			Duplicate points in output		
Dim.	BS	CDC	LMD	BS	CDC	LMD
2	0%	0%	0%	0%	68.8%	24.5%
3	0%	0%	0.4%	0%	82%	34.5%
4	0%	0%	1%	0%	85.5%	40.3%
5	3.6%	0.1%	3.9%	1.2%	82.1%	34.9%
6	19.7%	0.2%	16.5%	3.1%	74%	22.2%
7	38.8%	4.7%	39.5%	4.8%	60.9%	12.4%

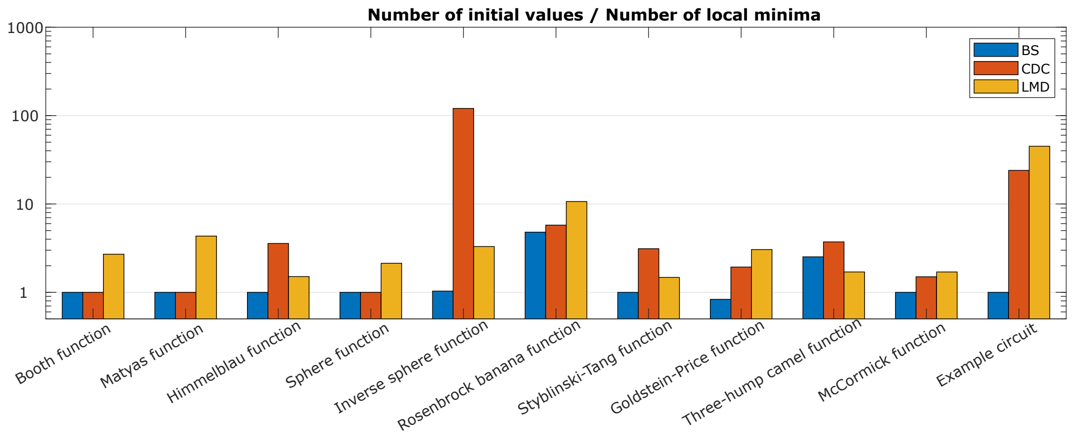


Fig. 3. Ratio between the number of initial values the algorithms found and the number of local minima of each function. Additionally, the analysis of the example circuit is shown for reference.

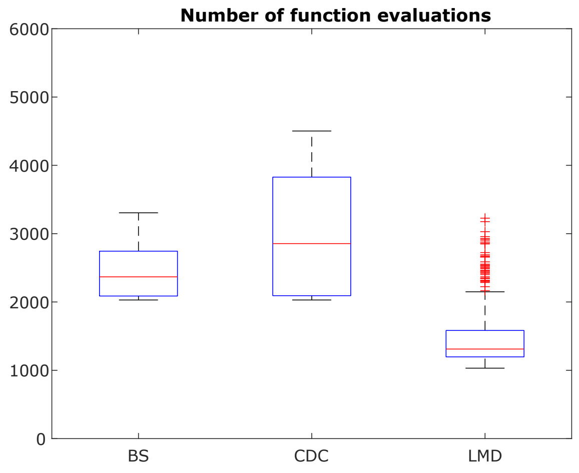


Fig. 4. Total number of function evaluations from initial value search to found optimum. The values contain the local optimum search too.

where $\mathbf{x}$ is an N-dimensional vector and x_i is the ith element of $\mathbf{x}$. This function has 2^N local minima at the corners of the search space.

Table I shows the ratio of missed local minima and the ratio of duplicate points per local minima. It can be seen that at most 4-dimensional problems, the BS and CDC algorithms did not miss any local minima in the search space. Over 5-dimensional functions, the algorithms started omitting regions of attraction. Out of three, the CDC algorithm missed the least number of local minima. In our experiments we have found that if the example function has at most 4 dimensions, the BS algorithm returns exactly the same number of initial values as the number of real local minima, while the other two algorithms always have multiple initial points in the same attraction regions. This problem is the most severe for the CDC algorithm: for example, even at the two-dimensional case, CDC returns more than three times as much duplicate points as real local minima. The LMD algorithm also produces a large number of duplicates, even below 5-dimensional problems too.

Based on our experiments, we suggest to use the BS algorithm for searching for initial values in numerical optimization based worst-case circuit analysis tasks.

VII. Conclusion and Further Research

Numerical optimization is a powerful tool in the worst-case analysis of complex circuits. The algorithms shown in this paper provide initial value sets suitable for global optimization tasks.

Further research include evaluating and comparing the performance of additional global optimization methods (e.g. GLOBAL [5] and memetic algorithms [4]).

VIII. Acknowledgement

Project no. 2019-1.3.1-KK-2019-00004 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2019-1.3.1-KK funding scheme.

References

[1] W. Smith, "Worst case circuit analysis-an overview (electronic parts/circuits tolerance analysis)," in *Proceedings of 1996 Annual Reliability and Maintainability Symposium*, 1996, pp. 326–334.

[2] A. Singhee and R. A. Rutenbar, "Why quasi-monte carlo is better than monte carlo or latin hypercube sampling for statistical circuit analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 11, pp. 1763–1776, 2010.

[3] A. Lokanathan and J. B. Brockman, "Efficient worst case analysis of integrated circuits," in *Proceedings of the IEEE 1995 Custom Integrated Circuits Conference*. IEEE, 1995, pp. 237–240.

[4] F. Schoen and L. Tigli, "Efficient large scale global optimization through clustering-based population methods," *Computers & Operations Research*, vol. 127, p. 105165, 2021.

[5] B. Bánhelyi, T. Csendes, B. Lévai, L. Pál, and D. Zombori, *The GLOBAL Optimization Algorithm: Newly Updated with Java Implementation and Parallelization*. Springer, 2018.

[6] J. W. Chinneck, "Analyzing mathematical programs using mprobe," *Annals of Operations Research*, vol. 104, no. 1, pp. 33–48, 2001.

[7] ——, "Discovering the characteristics of mathematical programs via sampling," *Optimization Methods and Software*, vol. 17, no. 2, pp. 319–352, 2002.

[8] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimization problems," *arXiv preprint arXiv:1308.4008*, 2013.

Lazy Abstraction for Probabilistic Systems

Dániel Szekeres, István Majzik
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
szekeresdani@edu.bme.hu, majzik@mit.bme.hu

***Abstract*—Reliability analysis of complex safety-critical systems by probabilistic model checking is often hindered by state space explosion. Abstraction is one way to counteract this problem. In this paper, we adapt an existing lazy abstraction algorithm to the analysis of Markov Decision Process reliability models and prove its soundness.**

***Index Terms*—Probabilistic Model Checking, Markov Decision Processes, Lazy Abstraction**

I. Introduction

It is crucial to guarantee the reliable and safe operation of safety-critical systems such as railway interlocking systems or embedded controllers in vehicles.

Probabilistic model checking is an automatic approach for the analysis of such quantitative properties with formal mathematical guarantees [1]. In this paper, we focus on the most fundamental task of safety-related probabilistic model checking: computing the probability of reaching an error state. If the behavior of the system under analysis includes non-deterministic aspects, we are interested in worst-case analysis: the probability of reaching an error state if the non-determinism is resolved so that it maximizes this probability. A Markov Decision Process (MDP) is able to describe the behavior of the system if discrete-time analysis and a discrete state space is sufficient, and the analysis of other modeling formalisms (like Markov Automata or Probabibilistic Timed Automata) can often be reduced to MDP analysis as well.

Practical usage of probabilistic model checking is hindered by the state-space explosion problem: the state-space of the MDP is often intractably large, which can be caused for example by concurrent execution of components that depend on each other. Abstraction is a way to counteract this problem, and several abstraction-based techniques have been adapted to probabilistic systems, like CEGAR [2], [3] and abstract interpretation [4]. Lazy abstraction is another approach that merges abstraction and refinement steps and applies refinement during state-space exploration only for those abstract states where it is necessary. It has seen success in non-probabilistic model checking, but it has yet to find its way into the probabilistic setting. In this paper, we adapt a lazy abstraction algorithm to the analysis of MDP models and prove its soundness.

Project no. 2019-1.3.1-KK-2019-00004 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2019-1.3.1-KK funding scheme.

II. Background and notations

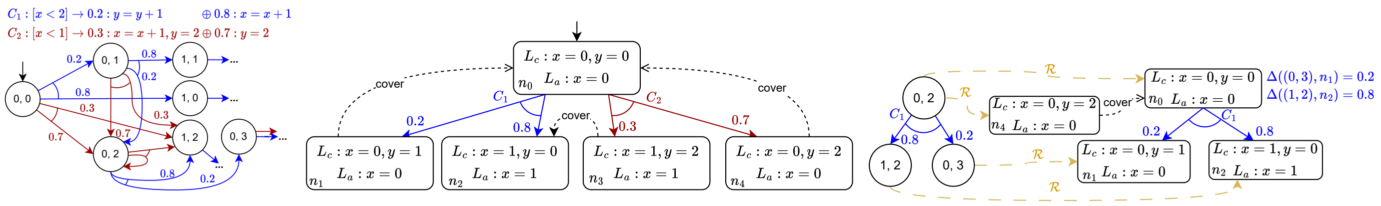
$\mathbb{D}(A)$ is the set of probability distributions over the set A. For $d \in \mathbb{D}(A), a \in A$, $d(a)$ denotes the probability measure of a according to d. $f : A \hookrightarrow B$ means that f is a *partial function* from A to B, and $Supp(f)$ denotes the set of values where f is defined. For $d \in \mathbb{D}(A)$, $Supp(d) = \{a \in A | d(a) > 0\}$. δ_x is a dirac distribution: $\delta_x(x) = 1, \forall y \neq x : \delta_x(y) = 0$.

a) Markov Decision Process (MDP): MDPs are low-level mathematical models able to describe probabilistic and non-deterministic behavior in discrete time. An MDP is a tuple $M = (S, Act, T, s_0)$, where S is the set of states, Act is the set of actions, $T : S \times Act \times S \to [0, 1]$ is the probabilistic transition function satisfying $\forall s \in S, a \in Act : \sum_{s' \in S} T(s, a, s') \in \{0, 1\}$ and $s_0 \in S$ is the initial state. An action $a \in Act$ is *enabled* in $s \in S$ if $\sum_{s' \in S} T(s, a, s') = 1$. If an action $a \in Act$ is enabled in $s \in S$, $T(a, s) \in \mathbb{D}(S)$ denotes the next state distribution after taking the action a in s, defined as $T(s, a)(s') = T(s, a, s')$. The intuitive behavior of an MDP is as follows: we start in s_0, then in each step, an action a is chosen non-deterministically from those enabled in the current state s, and the next state is sampled from $T(s, a)$. A *trace* of an MDP is an alternating list of states and actions $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \dots$ such that $T(s_{i-1}, a_i, s_i) > 0$ for each step. If we fix a strategy for resolving the non-determinism, the set of traces can be equipped with a probability measure: intuitively, the probability of a trace is the product of the probability of landing in each state of the trace after taking the action specified by the strategy in the previous state. For a detailed formal treatment, see e. g. [1].

Given an MDP M describing the system behavior and set of error states E, we want to compute (an upper approximation of) the probability measure of the set of traces that involve a state in E: $\mathbb{P}(\{(s_0 s_1 s_2 \dots) \mid \exists i \in \mathbb{Z}^+ : s_i \in E\})$. The result is the same if we make all error states absorbing – this way, we can restrict the analysis to finite traces.

b) Symbolic description of MDPs: Instead of explicitly constructing the state space of the model of a complex real-life system by hand, most such models are defined symbolically using state variables and operations on them.

We assume that the MDP to be analyzed is given by a set of state variables $\mathcal{V}$ and a set of probabilistic guarded commands $\mathcal{C}$. Each $v \in \mathcal{V}$ has an associated set $Range(v)$ of values it can take, and an initial value $v_0 \in Range(v)$. A *valuation* over $\mathcal{V}$ is a function $val : \mathcal{V} \to \bigcup_{v \in \mathcal{V}} Range(v)$ such that



(a) MDP given by two commands

(b) Unfinished ARG of the MDP with explicit value abstraction

(c) Visualization of the strong simulation relation

Fig. 1: Example of an MDP given through probabilistic guarded commands, an ARG of this MDP, and how the strong simulation maps from MDP states to ARG nodes

$\forall v \in \mathcal{V} : val(v) \in Range(v)$. The initial valuation of the model is a valuation val_0 such that $\forall v \in \mathcal{V} : val_0(v) = v_0$. The state space of the symbolically described MDP is a subset of $VAL_{\mathcal{V}}$, and its initial state distribution is δ_{val_0}.

Let $VAL_{\mathcal{V}}$ denote the set of all valuations over $\mathcal{V}$, $\mathcal{B}_{\mathcal{V}}$ the set of all Boolean expressions over $\mathcal{V}$, $\mathcal{E}^v_{\mathcal{V}}$ the set of all expressions over $\mathcal{V}$ that result in an element of $Range(v)$, and $\mathcal{E}_{\mathcal{V}} = \bigcup_{v \in \mathcal{V}} \mathcal{E}^v_{\mathcal{V}}$. A *next state assignment* is a function $n : \mathcal{V} \to \mathcal{E}_{\mathcal{V}}$, such that $\forall v \in \mathcal{V} : n(v) \in \mathcal{E}^v_{\mathcal{V}}$. Let $\mathcal{N}_{\mathcal{V}}$ denote the set of all next state assignments for $\mathcal{V}$. $eval(e, val)$ for $e \in \mathcal{E}_{\mathcal{V}}$ and $val \in VAL_{\mathcal{V}}$ is the constant resulting from replacing each $v \in \mathcal{V}$ in e with $val(v)$, then evaluating the (now constant) expression. $eval(n, val)$ for $n \in \mathcal{N}_{\mathcal{V}}$ is a valuation val' such that $\forall v \in \mathcal{V} : val'(v) = eval(n(v), val)$. $eval(d, val)$ for $d \in \mathbb{D}(\mathcal{N}_{\mathcal{V}})$ is the distribution $d' \in \mathbb{D}(VAL_{\mathcal{V}})$ such that $d'(val') = \sum_{n \in Supp(d) | eval(n,val) = val'} d(n)$.

A command $c \in \mathcal{C}$ consists of a guard $g_c \in \mathcal{B}_{\mathcal{V}}$ and a result distribution $d_c \in \mathbb{D}(\mathcal{N}_{\mathcal{V}})$. c is *enabled by* the valuation val iff $eval(g_c, val) = True$. The set of enabled actions in each state val of the resulting MDP consists of the set of commands enabled by val, and taking the command c results in the distribution $eval(d_c, val)$. An example MDP can be seen in Figure 1a. Widely used symbolic MDP description formats, like that of PRISM [5] or the JANI [6] interchange format can be mapped to this low-level symbolic description.

c) Abstraction: Abstraction is a successful approach to combat state-space explosion by ignoring information that is not needed to prove or disprove the property of interest. An *abstract model* is created which conservatively approximates the original *concrete model*: if the abstract model satisfies the property of interest, then the concrete model does as well.

Abstract states are commonly described symbolically using an *abstract domain*. Given a set of concrete states S, an abstract domain $D = (\hat{S}, \preceq, \alpha, \gamma)$ consists of the set of abstract states $\hat{S}$, a partial ordering relation $\preceq \subseteq \hat{S} \times \hat{S}$, an abstraction function $\alpha : 2^S \to \hat{S}$ and a concretization function $\gamma : \hat{S} \to 2^S$. α and γ form a Galois connection between the posets $(2^S, \subseteq)$ and $(\hat{S}, \preceq)$, meaning $\forall A \in 2^S, \hat{a} \in \hat{S} : \alpha(A) \preceq \hat{a} \iff A \subseteq \gamma(\hat{a})$. γ lets us treat abstract states as sets of concrete states; we write "$s \in \hat{s}$" for $s \in \gamma(\hat{s})$ when γ is clear from the context. $x \preceq y$ denotes $(x, y) \in \preceq$. $\hat{S}$ has two special elements: $\top$ and $\bot$ satisfying $\gamma(\top) = S, \gamma(\bot) = \{\}$.

For checking safety properties in the qualitative case, the conservative direction is to overapproximate the set of reachable states: a transition from $\hat{s} \in \hat{S}$ to $\hat{s}' \in \hat{S}$ is possible iff $\exists s \in \hat{s}, s' \in \hat{s}'$: a transition exists from s to s'. For the probabilistic setting, this changes to the requirement that the *probability* of reaching an error state in the abstract model must be *at least as high* as in the concrete one, which we will prove for the proposed algorithm.

The lazy abstraction algorithm needs an abstract domain for $S = VAL_{\mathcal{V}}$, for which we assume the existence of some operations. The algorithms in this paper are domain-agnostic as long as these can be computed. As an example, the explicit value domain satisfies these requirements.

For $a \in \mathcal{N}_{\mathcal{V}}$ and $\hat{s} \in \hat{S}$, $eval(a, \hat{s}) \in \hat{S}$ is the *abstract post operator* computing the result of applying a next state assignment in the abstract state space: $eval(a, \hat{s}) = \alpha(\{eval(a, s) | s \in \hat{s}\})$. For $b \in \mathcal{B}_{\mathcal{V}}$, $eval(b, \hat{s}) \in \{True, False, Unknown\}$ denotes evaluating b in the abstract state space: $True$ if b evaluates to $True$ for all $s \in \hat{s}$, $False$ if b evaluates to $False$ for all $s \in \hat{s}$, otherwise $Unknown$.

We assume the existence of a *block* operation: for an abstract state $\hat{s} \in \hat{S}$, a Boolean expression $b \in \mathcal{B}_{\mathcal{V}}$ and a concrete state $s \in \hat{s}$ such that $eval(b, s) = False$, $\hat{s}' = block(\hat{s}, b, s)$ is an abstract state satisfying $\hat{s}' \preceq \hat{s}, s \in \hat{s}', eval(b, \hat{s}') = False$. The goal of the operation is to compute a new abstract state by removing at least those states from $\hat{s}$ which satisfy b (potentially others as well, when the domain is not granular enough) while making sure to keep s.

We need to be able to represent the abstract states as Boolean expressions in the sense that for each $\hat{s} \in \hat{S}$ a $b_{\hat{s}} \in \mathcal{B}_{\mathcal{V}}$ must exist such that $\forall s \in S : eval(b_{\hat{s}}, s) = True \iff s \in \hat{s}$.

A successful approach to implement abstraction-based model checking is using abstraction-refinement methods: starting from a very coarse abstraction and introducing further information as the verification progresses until we can prove satisfaction or violation of the requirement.

d) Lazy abstraction: The main idea behind lazy abstraction is on-demand refinement of the abstract states during abstract state space exploration. We selected the lazy abstraction method described in [7] to apply to MDP model checking.

It builds an *Abstract Reachability Graph (ARG)* with each node labeled by both a concrete state and an abstract state: the concrete state is assumed to be able to represent all other states in the abstract state with respect to the enabled actions until this is disproven. The abstract labels are originally very coarse and are refined when needed. *Covering edges* are used to denote that the traces starting from a node cover all possible traces from another node – paths from the covered need not be explored. When an action is enabled in at least one element of the abstract label of a node, but not in the concrete label, the abstract label is *strengthened*: states where the action is enabled are blocked out of it. This operation can trigger other strengthenings.

If all enabled actions in all non-covered ARG nodes have been explored, the algorithm stops. The ARG overapproximates the concrete state space: for all reachable concrete states, there is a node whose abstract label contains it.

III. Application of the lazy algorithm to MDPs

In this section we adapt the lazy algorithm to analyzing a symbolic MDP given by a variable set $\mathcal{V}$ and a command set $\mathcal{C}_0$. Given an error state formula $\phi \in \mathcal{E}_{\mathcal{V}}$, the goal of the analysis is to compute the probability of reaching a state s where $eval(\phi, s) = True$. We show that the abstraction is sound for safety properties in the sense that the error probability in the abstract model overapproximates the concrete probability.

We assume that an abstract domain $(\hat{S}, \preceq, \alpha, \gamma)$ has been selected. The set of commands is extended with an error command: $\mathcal{C} = \mathcal{C}_0 \cup \{(\phi, \delta_{id})\}$, where id is an assignment that does not change any variable. This ensures that the finished ARG contains a node labeled with an error state exactly if an error state is reachable in the concrete state space.

The lazy abstraction algorithm explores the abstract state space by building an ARG: a tuple (N, E_T, E_C, L_c, L_a), where N is a set of nodes, $E_T \subseteq N \times \mathcal{C} \times \mathbb{D}(N)$ is a set of transition "edges" from nodes to node distributions labeled with commands, $E_C \subseteq N \times N$ is a set of directed *covering* edges, $L_c : N \to VAL_{\mathcal{V}}$ is the *concrete labeling function*, $L_a : N \to \hat{S}$ is the *abstract labeling function*.

The ARG is *well-formed* if it satisfies the following criteria: **a1)** abstract label contains the concrete label: $\forall n \in N : L_c(n) \in L_a(n)$, **a2)** concrete label can represent the whole abstract label with respect to the enabled commands: $\forall n \in N : \forall c \in \mathcal{C} : eval(g_c, L_c(n)) = False \implies eval(g_c, L_a(n)) = False$, **b1)** commands labeling a transition edge are enabled in the concrete state of the source: $\forall (n, c, \cdot) \in E_T : eval(g_c, L_c(n)) = True$, **b2)** there is a probability-preserving bijection f between nodes at the end of a transition edge and the assignments of the command labeling the edge which respects the result of the assignment (further explained below): $\forall (n, c, d_e) \in E_T : \exists (f : Supp(d_c) \longleftrightarrow Supp(d_e)) : \forall a \in Supp(d_c) : eval(a, L_c(n)) = L_c(f(a)) \wedge d_c(a) = d_e(f(a)) \wedge eval(a, L_a(n)) \preceq L_a(f(a))$, **c1)** abstract label of covering node must contain concrete label of covered node: $\forall (n, n') \in E_C : L_c(n) \in L_a(n')$, **c2)** covering node must be at least as abstract as the covered node: $\forall (n, n') \in E_C : L_a(n) \preceq L_a(n')$. **c3)** covering node cannot be covered: $\forall (n, n') \in E_C : \neg \exists (n', n'') \in E_C$.

An example ARG can be seen in Figure 1b. To make **b2)** easier to understand, let us see how it is satisfied in this example for the C_1 edge of node n_0. Intuitively, it means that the distribution at the end of the edge results from applying the command to the source node of the edge. The nodes at the end of this edge and their probabilities are n_1 with 0.2 and n_2 with 0.8. The assignments of this command are A_1 assigning $y+1$ to y and x to x, and A_2 assigning $x+1$ to x and y to y. The probability of A_1 is 0.2, that of A_2 is 0.8. The bijection f that makes *b2)* satisfied maps A_1 to n_1 and A_2 to n_2. Considering first A_1, this mapping is probability preserving: $d_c(A_1) = d_e(n_1) = 0.2$, it respects the assignments for the concrete labels: $eval(A_1, L_c(n_0)) = L_c(n_1)$, and also for the abstract labels: $eval(A_1, L_a(n_0)) \preceq L_a(n_1)$. Each of these also holds for A_2. In the lazy abstraction algorithm, the abstract label of a node can be coarser than the exact result of applying the post operator to its ancestor as long as all other constraints are satisfied.

The algorithm starts with an ARG with a single node $n_0, L_c(n_0) = val_0, L_a(n_0) = \top$. We extend this to a well-formed ARG where all nodes are either covered or fully expanded using expansion, covering and strengthening operations. If a node $n \in N$ is selected for expansion, we check for each $c \in \mathcal{C}$ whether $eval(g_c, L_c(n)) = True$. If so, a new node n'_i is created for each $a_i \in Supp(d_c)$ with $L_c(n'_i) = eval(a_i, L_c(n)), L_a(n'_i) = \top$, and a transition edge (n, c, d_e) is created such that $d_e(n'_i) = d_c(a_i)$ for $i = 1 \ldots |Supp(d_c)|$. For each newly created node n', we check whether $\exists n_c \neq n' \in N : L_c(n') \in L_a(n_c)$ such that n_c is not covered, and if so, a covering edge (n', n_c) is created for such an n_c and $L_a(n')$ is strengthened. If this leads to a violation of well-formedness constraint **b2)**, then n has to be strengthened as well.

If $eval(g_c, L_c(n)) = False$, then we compute $eval(g_c, L_a(n))$. If *False*, we move on to the next command or node. However, if it is true or *Unknown*, then **a2)** is violated, and $L_a(n)$ needs to be strengthened.

Whenever the value of $L_a(n)$ is changed for some node $n \in N$, the well-formedness constraints can be violated. If **c1)** is violated, the problematic covering edge is simply removed from E_C. Other constraints can then be restored by *strengthening* the abstract labels of nodes related to n: $L_a(n)$ is replaced by some $\hat{s}'$ such that $\hat{s}' \preceq L_a(n)$ and at least one well-formedness violation is eliminated.

If **a2)** is violated by a node n because of a command c that is enabled somewhere in $L_a(n)$ but not in $L_c(n)$, a new abstract label can be computed as $\hat{s}' = block(L_a(n), g_c, L_c(n))$. Because of the contract of *block*, $eval(g_c, \hat{s}') = False$, so this command no longer causes a constraint violation.

If **c2)** is violated by some covering edge $(n, n') \in E_C$, but **c1)** still holds for these nodes, then the current $L_a(n)$ must be replaced with $\hat{s}'$ such that $L_c(n) \in \hat{s}'$, $\hat{s}' \preceq L_a(n')$ and $\hat{s}' \preceq L_a(n)$ (referring to the current L_a). By describing $L_a(n')$ as a Boolean expression b_c, we can compute an appropriate $\hat{s}'$ as $block(L_a(n), \neg b_c, L_c(n))$.

Now assume that **b2)** is violated by an edge (n, c, d). Because of how the ARG is constructed, this means that a bijection f mentioned in the constraint could be constructed if we ignored the constraints on L_a, but the L_a part is violated by some state. Let us construct such a bijection f ignoring the L_a part. There exists an $a \in Supp(d_c)$ such that $eval(a, L_a(n)) \not\preceq L_a(f(a))$. By representing $eval^{-1}(a, L_a(f(a)))$ as a Boolean expression b and changing $L_a(n)$ to $block(L_a(n), b, L_c(n))$, the violation caused by the assignment a is eliminated. As the abstract label of n became finer, this might trigger another strengthening, which can propagate back up until the root node. Efficient implementations of the algorithm can use sequence interpolation to compute the strengthening of the whole path up to the root at once [7].

Strengthenings can create new violations triggering another strengthening, but all violations are eliminated after finite steps, and we continue expanding non-covered nodes.

The finished ARG can be treated as an MDP $(N, \mathcal{C}, T_{ARG}, n_0)$. Its transition function T_{ARG} is defined as follows. For a non-covered node n, a command $c \in C$ is enabled if there is an edge $(n, c, d) \in E_T$ in the ARG, and $T_{ARG}(n, c) = d$. For a covered node n, let n' denote its covering node (so $(n, n') \in E_C$). A command $c \in C$ is enabled in n iff it is enabled in n', and $T_{ARG}(n, c) = T_{ARG}(n', c)$. As the covering node cannot be covered, T_{ARG} is a well-defined function. Intuitively, we merge covered nodes into their covering nodes.

We prove the soundness of this abstraction using the notion of *strong simulation* of MDPs, similarly to how the soundness of another MDP abstraction algorithm was proven in [8]. It was shown in [9], that strong simulation preserves the safety subset of Probabilistic Computation Tree Logic (PCTL) properties [9], [10], meaning that whenever a strong simulation relation exists from an MDP M_1 to another MDP M_2, then the satisfaction of a safe PCTL property ϕ for M_2 implies the satisfaction of ϕ for M_1. As probabilistic reachability properties are a subset of safe PCTL, the soundness of our algorithm can be proven by constructing a strong simulation relation $\mathcal{R} \subseteq S \times N$ from the original MDP to the ARG.

Given two distributions $d^1 \in \mathbb{D}(S^1), d^2 \in \mathbb{D}(S^2)$ and a relation $\mathcal{R} \subseteq S^1 \times S^2$, $\Delta : S^1 \times S^2 \to [0, 1]$ is a *weight function* if: $\Delta(s^1, s^2) > 0$ implies $s^1 \mathcal{R} s^2$, $\forall s^1 \in S^1 : \sum_{s^2 \in S^2} \Delta(s^1, s^2) = d^1(s^1)$ and $\forall s^2 \in S^2 : \sum_{s^1 \in S^1} \Delta(s^1, s^2) = d^2(s^2)$. The existence of a weight function between d^1 and d^2 for a relation $\mathcal{R}$ will be denoted $d^1 \sqsubseteq_{\mathcal{R}} d^2$. Given two MDPs (S^1, Act, T^1, s^1_0) and (S^2, Act, T^2, s^2_0), a relation $\mathcal{R} \subseteq S^1 \times S^2$ is a *strong simulation relation* if **i)** $s^1_0 \mathcal{R} s^2_0$, **ii)** $\forall (s^1, s^2) \in \mathcal{R} : \forall a \in Act$ enabled in s^1 : a is enabled in $s^2 \wedge T^1(s^1, a) \sqsubseteq_{\mathcal{R}} T^2(s^2, a)$.

The relation that we use is given by $\mathcal{R} = \{(s, n) | s \in L_a(n)\}$. $s_0 \mathcal{R} n_0$, as $L_c(n_0) = s_0$ implies $s_0 \in L_a(n_0)$, so the starting states are related, **i)** holds. For all $(s, n) \in \mathcal{R}$ if n is not covered, then from $s \in L_a(n)$, **a2)** ($L_c(n)$ represents the whole $L_a(n)$) and the finishedness of the ARG (there is a transition edge labeled with c for each command enabled in $L_c(n)$) follows that all commands enabled in s are also enabled in n. A similar reasoning holds if n is covered by n', with the addition that $s \in L_a(n')$ because of **c2)**. Let s, n and c be any fixed concrete state, ARG node and command respectively from now such that $(s, n) \in \mathcal{R}$ and $c \in \mathcal{C}$ is enabled in s. Now we need to show the existence of weight function Δ between $eval(d_c, s)$ and $T_{ARG}(n, c)$. There is an edge $(n, c, d) \in E_T$ if n is not covered, and $(n', c, d) \in E_T$ if n' covers n. Let f be a bijection mapping the assignments of c to $Supp(d)$ that makes **b2)** true for this edge. Let $s_i = eval(a_i, s)$ and $d_i = d_c(a_i)$ for each assignment a_i of c. As $s \in L_a(n)$, $s_i \in eval(a_i, L_a(n))$. If n is not covered, we have from **b2)** that $eval(a_i, L_a(n)) \preceq L_a(f(a_i))$, so $s_i \in L_a(f(a_i))$. If $(n, n') \in L_C$, $L_a(n) \preceq L_a(n')$ from **c2)** implies $eval(a_i, L_a(n)) \preceq eval(a_i, L_a(n')) \preceq L_a(f(n))$, so $s_i \in L_a(f(n))$ in this case as well. This means that $s_i \mathcal{R} f(a_i)$ for each i. Now let us set $\Delta(s_i, f(a_i)) = d_i$ for each i, and let Δ be zero everywhere else. As $s_i \mathcal{R} f(a_i)$, the first condition of weight functions is satisfied. The second and third conditions are satisfied as the only non-zero element of the sum in both cases is exactly $d_i = eval(c, s)(s_i) = T_{ARG}(n, c)(n_i)$. This proves the existence of a valid weight function, which completes the proof of $\mathcal{R}$ being a strong simulation relation. From Thm. 19 in [9], it follows that if a probabilistic reachability property holds in the ARG, it also holds in the original MDP, which completes our proof. Figure 1c visualizes the relation.

As this relation is not a *bi*simulation, a problem with this abstraction is that the computed error probability on the ARG is only an overapproximation, and we have no control over how close it is to the concrete model. To make more precise error probability computation possible, we need to introduce an option to refine the ARG further on demand. We aim to address this issue in our future work.

IV. Conclusions

We have shown that the lazy abstraction scheme of [7] is applicable to symbolic MDPs and gave a formal proof of its soundness. We have shown a shortcoming of the direct adaptation in that it cannot be further refined. We plan to address this issue in the future by developing a variant based on stochastic games. This work is currently in its theoretical phase, implementation of the proposed algorithm is in progress in the Theta model checking framework.

References

[1] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic model checking," in *SFM'07*, ser. LNCS, vol. 4486. Springer, 2007, pp. 220–270.

[2] M. Kattenbelt, M. Kwiatkowska *et al.*, "Abstraction refinement for probabilistic software," in *VMCAI'09*, 2009.

[3] H. Hermanns, B. Wachter, and L. Zhang, "Probabilistic CEGAR," in *CAV'08*, ser. LNCS, vol. 5123. Springer, 2008, pp. 162–175.

[4] P. Cousot and M. Monerau, "Probabilistic abstract interpretation," in *ESOP'12*, ser. LNCS, vol. 7211. Springer, 2012, pp. 169–193.

[5] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *CAV'11*, 2011.

[6] C. E. Budde, C. Dehnert *et al.*, "Jani: Quantitative model and tool interaction," in *TACAS'17*, ser. LNCS, vol. 10206, 2017, pp. 151–168.

[7] T. Tóth and I. Majzik, "Configurable verification of timed automata with discrete variables," *Acta Informatica 59*, pp. 1–35, 2022.

[8] B. Wachter, L. Zhang, and H. Hermanns, "Probabilistic model checking modulo theories," in *QEST'07*. IEEE Computer Society, 2007, pp. 129–140.

[9] R. Segala and N. A. Lynch, "Probabilistic simulations for probabilistic processes," in *CONCUR '94*, ser. Lecture Notes in Computer Science, B. Jonsson and J. Parrow, Eds., vol. 836. Springer, 1994, pp. 481–496.

[10] C. Baier, J. Katoen *et al.*, "Comparative branching-time semantics for Markov chains," *Inf. Comput.*, vol. 200, no. 2, pp. 149–214, 2005.

Improving the Convergence Speed of the Resonator-Based Observer by Significant Component Selection

András Palkó, László Sujbert
Budapest University of Technology and Economics
Department of Measurement and Information Systems
Műegyetem rkp. 3., H-1111 Budapest, Hungary
Email: {palko, sujbert}@mit.bme.hu

***Abstract*—The resonator-based observer has been developed for measuring the harmonic components of a periodic signal with known fundamental frequency. In certain applications, the signal to be processed is sparse in the frequency domain: a subset of its harmonic components have negligible amplitude. This paper presents some extensions of the resonator-based observer which can exploit this sparsity to speed up the convergence. The performance of the proposed structures is demonstrated by simulation examples.**

***Index Terms*—periodic signals, order tracking, sparsity, convergence speed**

I. Introduction

In many applications, periodic signals are analyzed. Measuring their harmonic components is also known as order tracking [1]. Examples are active noise control, vibration analysis of rotating machines or line voltage harmonic analysis.

This problem can be approached in a model-based way. When the fundamental frequency is known and constant, the resonator-based observer (RBO) [2] is an adequate solution. The basic RBO has been already extended in multiple different ways. For unknown or changing fundamental frequency, the Adaptive Fourier Analyzer has been developed [3]. Another extension is the ability to handle missing samples [4].

Generally, a signal is said to be sparse if there is a basis in which it can be described with only a few nonzero coefficients. The notion of sparsity can be applied to periodic signals as well: strictly, it would mean that the majority of the Fourier coefficients are zero. In this paper, we will be more concessive: by sparse we mean that a non-negligible subset of the coefficients have (approximately) zero amplitude.

Trivial examples for sparse periodic signals are a pure sine wave, or a square wave with 50% duty cycle. A more practical example is the vibration caused by a ventilator: e.g. if it has five blades, then the 5th, 10th, 15th, …components are expected to have significantly more power.

In this paper we present some extensions of the RBO which are able to exploit the sparsity of their periodic input signal in order to speed up the convergence. The core idea is to select those components which are presumably negligible, and exclude them from the main structure. The main characteristics of the structures are illustrated by simulations.

The structure of the paper is as follows: Section II reviews the RBO, while the proposed structures are presented in Section III. Section IV shows some examples, and the paper concludes in Section V.

Project no. 2019-1.3.1-KK-2019-00004 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2019-1.3.1-KK funding scheme.

II. Preliminaries

A. Conceptual Signal Model

The so-called conceptual signal model is the complex Fourier series of a periodic signal:

$$d = \sum_{k=-L}^{L} x_k \qquad x_k = X_k c_k \qquad c_k = \mathrm{e}^{\mathrm{j}2\pi f_1 k n} \tag{1}$$

for $k = -L, \ldots, L$, where X_k is the kth Fourier coefficient, x_k is the corresponding Fourier term, $\mathrm{j} = \sqrt{-1}$, f_1 is the fundamental frequency (relative to the sampling frequency), and n is the time index. In order to keep the notation clear, explicit time indices will not appear unless necessary.

Note carefully the difference between X_k and x_k. X_k is a Fourier coefficient, which is constant for a given periodic signal, while x_k is obtained by rotating X_k according to the frequency of the component and the time index.

The components are modeled up to the Nyquist frequency, thus $Lf_1 < 0.5 < (L+1)f_1$. The lack of modeled component at the Nyquist frequency poses no problem in practice.

Figure 1 depicts the conceptual signal model. The blue box is a block definition for further use. Each integrator also has a u_k update signal:

$$X_k(n+1) = X_k(n) + u_k(n) \tag{2}$$

B. Resonator-Based Observer

The RBO is obtained by designing a state observer for the conceptual signal model. As the state variables are the Fourier coefficients, the observer estimates them directly. The observer can be described by the following equations:

$$y = \sum_{k=-L}^{L} \hat{x}_k \quad \hat{x}_k = \hat{X}_k c_k \quad e = d - y \quad u_k = \frac{\alpha}{N} g_k e \tag{3}$$

$$g_k = \bar{c}_k = \mathrm{e}^{-\mathrm{j}2\pi f_1 k n} \quad (k = -L, \ldots, L) \tag{4}$$

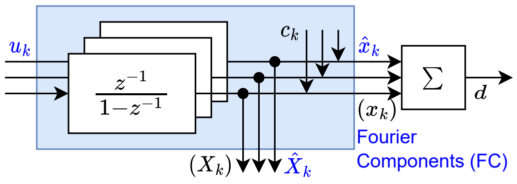


Fig. 1: The conceptual signal model. The blue box and its signals define the Fourier Components (FC) block and its interface for further figures. The equations in Section II-A use the X_k and x_k notation, all other equations and figures use the $\hat{X}_k$ and $\hat{x}_k$ notation.

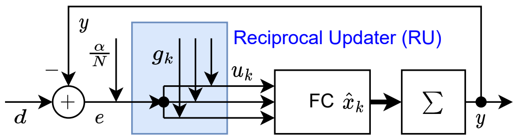


Fig. 2: The resonator-based observer. The blue box defines the Reciprocal Updater (RU) block for further figures. The thick line represents a vectorial signal. This convention is kept in further figures.

where y, e and d are the estimated input, error and input signals, respectively. $\hat{X}_k$ and $\hat{x}_k$ are the estimates of the kth Fourier coefficient and term, respectively, $0<\alpha\leq 1$, $N=2L+1$ is the number of modeled coefficients, g_k is a reciprocal complex exponential and $\bar{\cdot}$ denotes the complex conjugate.

The RBO is illustrated in Fig. 2. Here we define the Reciprocal Updater (RU) block. The modulation-demodulation scheme realized with c_k and g_k can falsely imply that the RBO is time variant. An equivalent formalization places the modulator-demodulator pair "inside" the integrator [5], which is clearly a time invariant system. These two equivalent formalizations also exist for the proposed structures.

In steady-state, the estimated and the original Fourier-coefficients are equal, thus the signal is perfectly reconstructed. The observer provides unbiased estimates of the Fourier coefficients [2].

If $0<\alpha<1$, the Fourier coefficient estimates are exponentially averaged [6] with an equivalent time constant of

$$\beta = 1-(1-\alpha)^{\frac{1}{N}} \tag{5}$$

We can see in (5) that for smaller N, the settling is faster, since there are less parameters to adapt.

For β small enough (which is usually granted in practical cases) the RBO is able to work over an arbitrary frequency set with similar convergence characteristics [6]. The "rows" in Fig. 2 (from u_k to $\hat{x}_k$ for a given k) are also called channels. Each channel corresponds to a single harmonic component. The magnitude response of any channel (from d to $\hat{x}_k$) is [6]

$$|H_k(f)| \begin{cases} =1 & \text{at the own frequency of the channel} \\ =0 & \text{at the frequencies of other channels} \\ >0 & \text{at any other frequency} \end{cases} \tag{6}$$

These relationships are illustrated for two cases in Fig. 3 (for $\hat{x}_2$). The blue line corresponds to a full RBO with $f_1 = \frac{1}{15}$, $\alpha = 1$. The 3rd and 6th components have been removed from this structure for the red line.

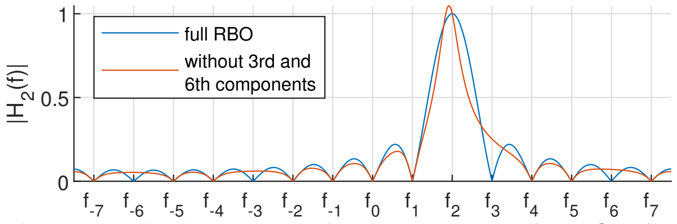
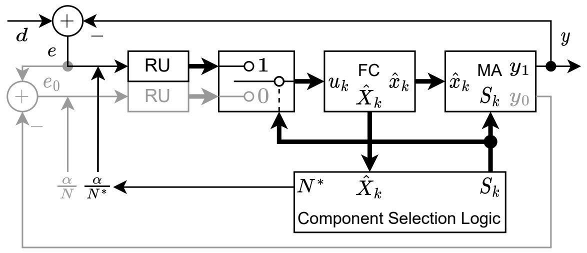


Fig. 3: Magnitude response of a single channel (from d to $\hat{x}_2$, on the interval $[-f_s/2, f_s/2]$). Blue line: full RBO with 15 channels. Red line: RBO with arbitrary frequencies.



d
e
e_0
y
RU
RU
1
0
FC u_k $\hat{X}_k$ $\hat{x}_k$
MA $\hat{x}_k$ y_1 S_k y_0
$\frac{\alpha}{N}$
$\frac{\alpha}{N^*}$
N^* $\hat{X}_k$ S_k
Component Selection Logic

Fig. 4: RBO with significant component selection. The main loop is drawn with black, while the auxiliary loop is depicted in gray color.

III. The Proposed Structures

A. Concept

Let us consider a strictly sparse periodic signal and process it using the original RBO. In steady-state, there will be channels whose state variable will be zero. Removing them would have no effect on the estimated signal.

During the settling (with $0 < \alpha < 1$), each coefficient estimate (approximately) exponentially tends to their steady-state value. Assuming arbitrary initial state, after some time the magnitudes of the nonzero (zero) coefficient estimates will be significant (negligible).

The idea is to automatically distinguish between the significant and the negligible coefficients and their corresponding channels. Since the input can be described with only the significant coefficients, keep only them and drop the negligible ones from the main adaptation loop. Consequently, the main loop will contain less channels, which results in a faster convergence.

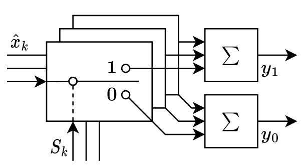
Moreover, it is conceivable that over time, the coefficients of the input signal change, some zeros become nonzeros or vice versa. Thus, the negligible components should not be dropped totally from the structure, but placed in an auxiliary loop and adapted there. If any of them becomes large enough, they can be placed back into the main loop.

B. Formalization

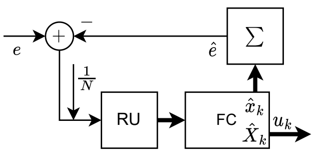
The basic proposed structure (Fig. 4, it will be referred as CSL+RU) can be described by formalizing the above ideas. There are two new blocks: the Component Selection Logic (CSL) and the Multiplexed Adder (MA).

Let us define the selection indicator for $k=-L,\ldots,L$:

$$S_k = \begin{cases} 1 & \text{if the } k\text{th component is significant} \\ 0 & \text{if the } k\text{th component is negligible} \end{cases} \tag{7}$$



(a) Multiplexed Adder (MA)



(b) Error Decomposition Updater (EDU)

Fig. 5: Block definitions for the proposed structures

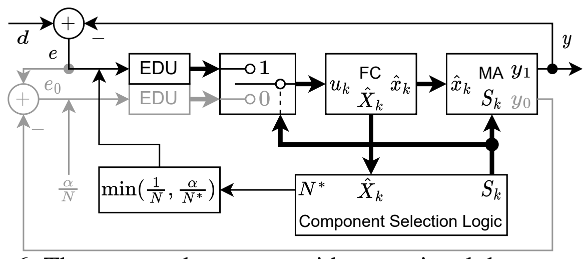


Fig. 6: The proposed structure with error signal decomposition

The Component Selection Logic provides this indicator along with the $N^* = \sum_{k=-L}^{k=L} S_k$ number of selected components.

An MA (Fig. 5a) is used to calculate the y and y_0 output signals of the main and auxiliary loops, respectively:

$$y = \sum_{k=-L}^{L} \hat{x}_k S_k \qquad y_0 = \sum_{k=-L}^{L} \hat{x}_k \left(1 - S_k\right) \tag{8}$$

The two error signals are

$$e = d - y \qquad e_0 = e - y_0 \tag{9}$$

The update signals of the two loops can be expressed in one equation:

$$u_k = \frac{\alpha}{N^*} S_k g_k e + \frac{\alpha}{N} \left(1 - S_k\right) g_k e_0 \tag{10}$$

Note that the definition of S_k implies that the state variables are updated separately.

C. Discussion

The main loop is an RBO with an arbitrary (but special) frequency set. Since $N^* \leq N$, the convergence is faster than that of the original RBO. If the user knew which components are significant, he could use a traditional RBO with frequencies set to those components. The main novelty of this approach is that it detects the orders of significant components without user interaction. Moreover, the increase in convergence speed is independent from the orders of the significant components. They can be arbitrarily grouped or scattered over the spectrum.

For noise suppression in the main loop, the results of [6] apply, with N^* instead of N in the formulas. The variance of a given significant coefficient is inversely proportional to N^* (this variance is not less than in the original RBO). As a consequence, the summed variance of the coefficients in the main loop (which is related to the variance of y by Parseval's theorem) is approximately independent of the sparsity. In other words, the noise bandwidth from d to y is the same as that of in the original RBO.

In Fig. 4 the auxiliary loop is drawn as running on the error signal. An equivalent point of view is that the output of the auxiliary loop contains all channels and its input is d, not e. This is the reason this loop uses N in the update equation.

D. Component Selection Logic

The responsibility of the CSL is to provide the selection indicator, i.e. distinguish between the significant and the negligible components. In this paper, we take a simple thresholding approach. This threshold can be a fixed value, given a priori, or it can be dynamic, e.g. based on the coefficient with maximal magnitude:

$$S_k = \begin{cases} 1 & \text{if } |\hat{X}_k| \geq \gamma |\hat{X}|_{\max} \\ 0 & \text{else} \end{cases} \tag{11}$$

where $0 < \gamma \ll 1$. Hysteresis can be used in order to eliminate the "juggling" of components between the two loops.

E. Error Signal Decomposition

Let us consider the case when besides the significant components, there are small (but not zero) ones in d, and they do not get selected. Alternatively, let us consider the effect of selection errors.

Let one not selected nonzero component be the ith one and let us examine its effect on the kth (selected) component. The magnitude response of the main loop has the characteristics outlined in (6). Consequently, $|H_k(f_i)| > 0$. This means that the not selected component at f_i causes an error in the measurement of the selected component at f_k.

For a particular example, let us take the case depicted with the red line in Fig. 3 and consider the effect of the not selected nonzero 3rd component on the selected 2nd one. Since $|H_2(f_3)| \approx 0.25 > 0$, this component at f_3 causes an error in the measurement of the selected component at f_2.

Thus the not selected nonzero components cause some error in the measurement of the selected components. As a result, the selected components do not vanish entirely from e. With a similar reasoning one can see that in this case the precision of the auxiliary loop is also impaired.

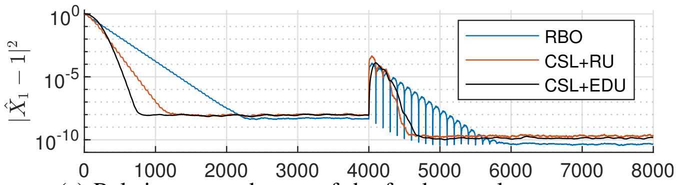
This problem can be solved via the decomposition of the error signal. The structure is modified slightly: instead of the RU blocks, an Error Decomposition Updater (EDU, Fig. 5b) is used in both loops (Fig. 6). The EDU is a full RBO run on e with $\alpha = 1$, and the Fourier coefficient estimates are taken as update signals. This variant will be referred as CSL+EDU.

Since the EDU is a full RBO, its magnitude response from e to u_k is characteristically same as the blue line in Fig. 3, regardless of the actual selection. Now $|H_k(f_i)| = 0 \, (i \neq k, i = -L, \ldots, L)$, thus the not selected nonzero components do not cause error in the measurement of the selected ones. For our particular example, the blue line in Fig. 3 has a zero at f_3, thus no component appears in u_2.

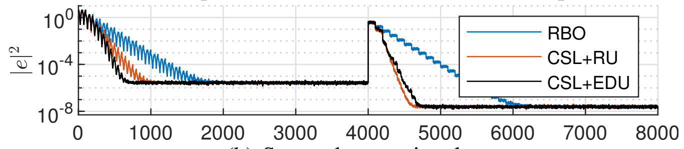
Since there is a new feedback loop inside the main loop, the $\frac{\alpha}{N^*}$ gain of the main loop cannot get as large as for CSL+RU. In our experience, an upper bound of $\frac{\alpha}{N^*} \leq \frac{1}{N}$ yields similar convergence to the CSL+RU.

IV. Examples

The examples model the measurement of the significant harmonic components of the line voltage. As such, the input signal will have 50 Hz frequency, sampled at 5 kHz. The fundamental component has a magnitude of 1 and there is no bias. The examples will differ in the higher harmonics.



(a) Relative squared error of the fundamental component



(b) Squared error signal

Fig. 7: Measurement errors by the time index for strictly sparse signals, averaged from 100 simulations

A. Strictly Sparse Signal

In the first example, the input signal is strictly sparse. In the beginning, the even harmonics are zeros, while the odd ones have a random magnitude drawn uniformly from $[0.05, 0.1]$ and a random phase. After 4000 samples, the harmonic components change: the odd ones become zeros and the even ones up to the 20th order get a random magnitude and phase the same way as the even ones before. There is an additive white Gaussian noise on the input, with 60 and 80 dB SNR in the two signal parts.

This signal is processed using the original RBO, CSL+RU and CSL+EDU. All structures use $\alpha = 0.35$. The CSLs are operated by (11) with $\gamma = 0.02$, 4 times per signal period.

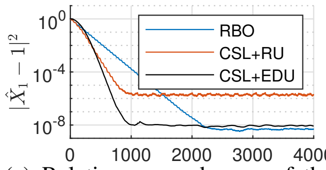
100 such simulations were conducted. The averaged relative squared error of the fundamental component is shown in Fig. 7a. Both proposed structures are able to speed up the convergence. Although there is an upper bound on the feedback gain of CSL+EDU, it was even faster than the CSL+RU. This acceleration depends also on the sparsity of the signal: the less components a signal has, the faster the convergence is for the proposed structures (the error of the proposed structures decays faster for the sparser signal part; the RBO is unaffected by the sparsity). The proposed structures have only slightly worse steady-state error than the RBO, due to their larger noise bandwidth.

The error signals (Fig. 7b) show the same convergence characteristics as the selected component estimates. Moreover, the steady-state reconstruction error is the same for all structures.

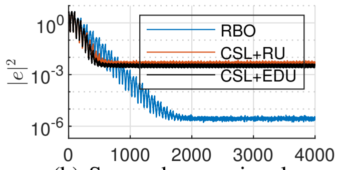
At the signal change point, the fundamental component is unchanged. As a result, its error jumps only because the other components affect it during the transient. After such an abrupt change in the Fourier coefficients, some time is needed for the CSL to actualize the selection (since the corresponding component estimates need to change). During the settling, some channels may be placed back and forth multiple times.

B. Effect of Small Components

The second example illustrates the effect of the small components. The input signal has the same parameters as in the first half of the previous example, with one exception: the small components are not zero, but have a magnitude drawn independently from a normal distribution with zero mean and



(a) Relative squared error of the fundamental component



(b) Squared error signal

Fig. 8: Measurement errors by the time index for for signals with significant and small components, averaged from 100 simulations.

0.01 standard deviation, and a random phase. The structures and their parameters are the same as before.

Again, results from 100 simulations were averaged to obtain the results. For the fundamental component estimate (Fig. 8a), CSL+RU has a significantly higher error than the other structures, the original RBO included: the not selected nonzero components cause an error in the selected ones. CSL+EDU has only slightly worse error than the RBO, due to its larger noise bandwidth. Moreover, as in the previous example, the proposed structures have faster convergence.

It is not surprising that CSL+RU has a higher steady-state reconstruction error than the RBO (Fig. 8b). But one could expect the CSL+EDU to have significantly lower error than the CSL+RU. This is unfounded though: even when all selected components are measured perfectly, the error signal of the main loop contains all the small components by design.

V. Conclusion

This paper presented two structures to improve the convergence speed of the RBO for sparse periodic signals. After reviewing the original RBO, the proposed structures have been presented. The core idea is to automatically separate the significant and negligible components, and measure only the significant ones in the main loop Little additional complexity is required: parallel and/or series connected resonators, adders, switches and some control logic. Since this way there are fewer parameters to adapt, the convergence becomes faster. The properties of the proposed structures were demonstrated with simulation examples: they converge faster than the original, and the speed depends on the sparsity of the input. Using error signal decomposition, the left out nonzero components cause no error in the measurement of the selected ones.

References

[1] A. Brandt, T. Lagö, K. Ahlin, and J. Tůma, “Main principles and limitations of current order tracking methods,” *Sound & vibration*, vol. 39, pp. 19–22, 03 2005.

[2] G. Peceli, “A common structure for recursive discrete transforms,” *IEEE Trans. Circuits Syst.*, vol. 33, no. 10, pp. 1035–1036, 1986.

[3] L. Sujbert, G. Simon, and G. Peceli, “An observer-based adaptive fourier analysis [tips & tricks],” *IEEE Signal Process. Mag.*, vol. 37, no. 4, pp. 134–143, 2020.

[4] G. Orosz, L. Sujbert, and G. Peceli, “Analysis of resonator-based harmonic estimation in the case of data loss,” *IEEE Trans. Instrum. Meas.*, vol. 62, no. 2, pp. 510–518, 2013.

[5] Z. Kollar, F. Plesznik, and S. Trumpf, “Observer-based recursive sliding discrete fourier transform [tips & tricks],” *IEEE Signal Process. Mag.*, vol. 35, no. 6, pp. 100–106, 2018.

[6] L. Sujbert, G. Peceli, and G. Simon, “Resonator-based nonparametric identification of linear systems,” *IEEE Trans. Instrum. Meas.*, vol. 54, no. 1, pp. 386–390, 2005.

Behavior of Observer Performing Discrete Gabor Transform

Bence Ország, László Sujbert
Department of Measurement and Information Systems,
Budapest University of Technology and Economics
Email: {orszag, sujbert}@mit.bme.hu

***Abstract*—The theory of discrete Gabor transforms provides a constructive and interpretable way of creating biorthogonal transforms. These transforms can achieve a non-uniform frequency resolution and they can be implemented recursively by observers based on Hostetter's approach. The dead-beat property of the observers were previously shown for coherent signals with the help of an adequately chosen conceptual signal model. This paper investigates the properties of the signal model-observer pair and derives the dead-beat property in the case of noncoherent signals based on a modified signal model.**

***Index Terms*—signal processing, Gabor transform, observer, dead-beat property**

I. Introduction

The usage of signal transforms is widespread in digital signal processing (DSP) algorithms due to their ability to emphasize and quantify relevant signal properties. In the case of the discrete Fourier transform (DFT) this is achieved by a purely frequency domain representation. In certain DSP methods this might prohibit the use of DFT due to the lack of time domain transient information.

To circumvent the issue one could use (the discrete variant of) the short-time Fourier transform (STFT) which, in essence, is the DFT of (possibly overlapping) segments of the original signal acquired by sliding a window through it. This leads to a so-called time-frequency representation. It is natural to ask what is the temporal and frequency resolution of this description. For an answer it is beneficial to interpret said transforms as filter banks because their temporal resolution is in connection with the decimation rates of the filters while the frequency resolution is described by their bands.

Another question that arises in connection with time-frequency representations is whether they provide sufficient information to reconstruct the original signal, phrased differently, whether the underlying transformation is invertible. The answer can be formulated with the toolset of frame theory [1] which is suitable to lay the foundations of nonstationary Gabor frames [2]. These provide the basis of transforms whose frequency resolution can be chosen almost arbitrarily (e.g. logarithmically), which is practical for audio applications. Based on these it is relatively straightforward to construct biorthogonal transforms [3] and this property enables a recursive implementation by observers based on Hostetter's approach [4]. The realization of the transforms is possible with filters, but they cannot reconstruct noncoherent signals while observers can which is proven by this paper.

Section II reviews the construction and specification of the relevant generalization of discrete Gabor transforms while in Section III the conceptual signal models and their observer and it's relevant properties are presented. Section IV contains the derivation of the dead-beat property in case of noncoherent signals while Section V concludes the paper.

II. Discrete Gabor Transform

Frames can be thought of as the set of (possibly linearly dependent) signals whose weighted sum could reproduce an arbitrary signal of a space. For example, in the case of the Fourier series these signals are complex exponentials and their linear combinations can reproduce periodic signals. The transform coefficients are exactly the weighting coefficients.

The frequency adaptive painless biorthogonal nonstationary discrete Gabor transforms [3] are an alternative to the DFT. These transforms can be chosen with near arbitrary frequency resolution unlike the DFT which has a linear one. In practical terms their construction is carried out based on the frequency domain specification of a finite impulse response (FIR) filter bank. The passbands of the filters cannot overlap based on their N point DFT, where N is the length of the FIR filters. If the number of the bands is L then $l = 0, \cdots, L-1$ a_l decimation parameters should be chosen satisfying

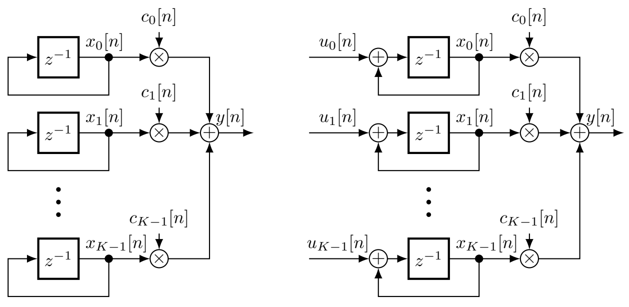
$$a_l \leq \frac{N}{N_l} \text{ and } \sum_{l=0}^{L-1} \frac{1}{a_l} = 1, \qquad (1)$$

where N_l is the length of the support of the respective passband. Furthermore a_l and N/a_l must be integer for all l.

Given that these conditions hold the resulting transform will be biorthogonal with the following reciprocal basis:

$$\mathbf{G} = \begin{pmatrix} \widetilde{\mathbf{g}}_0^T\langle 0 \cdot a_0\rangle \\ \widetilde{\mathbf{g}}_0^T\langle 1 \cdot a_0\rangle \\ \vdots \\ \widetilde{\mathbf{g}}_0^T\langle (N/a_0 - 1) \cdot a_0\rangle \\ \widetilde{\mathbf{g}}_1^T\langle 0 \cdot a_1\rangle \\ \vdots \\ \widetilde{\mathbf{g}}_{L-1}^T\langle (N/a_{L-1} - 1) \cdot a_{L-1}\rangle \end{pmatrix}, \qquad (2)$$

where $\widetilde{\mathbf{g}}_l \in \mathbb{C}^N$ are the impulse responses of the FIR filters and $\mathbf{v}\langle p\rangle$ denotes the circular shift of the $\mathbf{v}$ vector by p. The square matrix $\mathbf{G}$ represents the reciprocal basis, while the basis is the transpose of it's inverse due their biorthogonality.



(a) Coherent case. (b) Noncoherent case.

Fig. 1: Conceptual signal models.

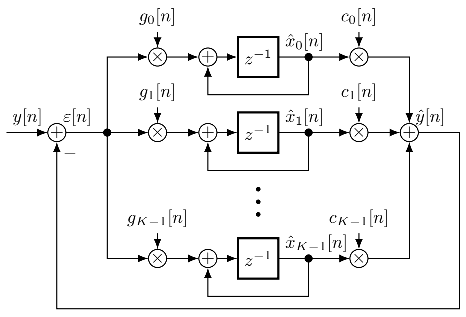


Fig. 2: An observer for recursive transformations.

III. A Common Structure for Recursive Discrete Transforms

A. Conceptual Signal Models

The conceptual signal model [4] is a linear system whose $y[n]$ output is acquired by the summation of $c_k[n]$ basis vectors weighted by $x_k[n]$ state variables. The total number of the basis vectors is K. They are periodic by definition with period N and their values are given on the $n = 0, \cdots, N-1$ timesteps so for any $n \in \mathbb{Z}$

$$c_k[n] = c_k[\mathrm{mod}\,(n, N)], \tag{3}$$

where $\mathrm{mod}\,(n, N)$ is the positive remainder of the integer division n/N. This periodic extension is assumed for the $\mathbf{g}[n]$ and $\mathbf{e}[n]$ signals introduced later in the article. In this setting with constant $\mathbf{x}[n]$ state variables it is possible to construct all discrete signals with period N with the choice of a biorthogonal basis which enforces the equality $N = K$. These periodic signals can be referred to as coherent signals and their conceptual signal model is shown on Fig. 1a.

Let's denote

$$\mathbf{x}[n] = \begin{pmatrix} x_0[n] & x_1[n] & \cdots & x_{K-1}[n] \end{pmatrix}^T, \tag{4}$$

$$\mathbf{c}[n] = \begin{pmatrix} c_0[n] & c_1[n] & \cdots & c_{K-1}[n] \end{pmatrix}^T. \tag{5}$$

This means that the output is

$$y[n] = \mathbf{c}^T[n]\mathbf{x}[n]. \tag{6}$$

Appendix A proves that arbitrary, noncoherent signals can be constructed if $\mathbf{x}[n]$ is not constant and gives a constructive way of choosing it's value to achieve a desired $y[n]$ output. But in order to modify $\mathbf{x}[n]$ the conceptual signal model must be extended with inputs shown on Fig. 1b. Assuming that $\mathbf{x}[n]$ is known for any $n \in \mathbb{Z}$ it is straightforward to construct an $\mathbf{u}[n]$ input which modifies $\mathbf{x}[n]$ accordingly:

$$\mathbf{u}[n] = \mathbf{x}[n+1] - \mathbf{x}[n] = \mathbf{g}[n+1]y[n+1] - \mathbf{g}[n]y[n], \tag{7}$$

where $\mathbf{g}[n]$ is the reciprocal basis introduced in the following section.

B. Observer

The state variables of the described signal model can be estimated by a properly designed observer which was introduced and analyzed in [4]. This can be seen in Fig. 2. It tries to reconstruct the $y[n]$ input signal by refining the $\hat{\mathbf{x}}[n]$ estimated state variables based on the $\varepsilon[n]$ reconstruction error with the help of the $g_k[n]$ reciprocal basis. The latter is denoted by

$$\mathbf{g}[n] = \begin{pmatrix} g_0[n] & g_1[n] & \cdots & g_{K-1}[n] \end{pmatrix}^T. \tag{8}$$

With the notation introduced so far, the timecourse of the estimated state variables can be given by the following equation:

$$\hat{\mathbf{x}}[n+1] = \hat{\mathbf{x}}[n] + \mathbf{g}[n]\mathbf{c}^T[n](\mathbf{x}[n] - \hat{\mathbf{x}}[n]). \tag{9}$$

It was proven in [4] that $\hat{\mathbf{x}}[n] = \mathbf{x}[0]$ after N time steps (or less) if $\mathbf{g}[n]$ and $\mathbf{c}[n]$ form a biorthogonal basis and the conceptual signal model is the one seen in Fig. 1a.

IV. Derivation of dead-beat property in case of noncoherent signals

A. Preliminaries

The basis can be collected into a matrix as row vectors:

$$\mathbf{C} = \begin{pmatrix} \mathbf{c}[0] & \mathbf{c}[1] & \cdots & \mathbf{c}[N-1] \end{pmatrix}. \tag{10}$$

This gives rise to an alternative notation with the help of the $\mathbf{e}[n]$ standard basis vectors

$$\mathbf{c}[n] = \mathbf{C}\mathbf{e}[n]. \tag{11}$$

This can be applied to the reciprocal basis as well which leads to a concise representation of the biorthogonality of the bases:

$$\mathbf{G}\mathbf{C}^T = \mathbf{I}. \tag{12}$$

B. Observability

The previous section reviewed the properties of the observer and stated that it is able to estimate the state variables without error. This is possible in the case of the extended conceptual signal model shown on Fig. 1b by proving that it is completely observable. Theorem 3 achieves exactly that, to apply it one needs to determine the canonical form of the model:

$$\mathbf{x}[n+1] = \mathbf{I}\mathbf{x}[n] + \mathbf{I}\mathbf{u}[n], \tag{13}$$

$$\mathbf{y}[n] = \mathbf{c}^T[n]\mathbf{x}[n]. \tag{14}$$

In this case the observability gramian defined in the appendix

$$\mathbf{O}[n_0] = \begin{pmatrix} \mathbf{c}^T[n_0] \\ \mathbf{c}^T[n_0+1] \\ \vdots \\ \mathbf{c}^T[n_0+N-1] \end{pmatrix} \tag{15}$$

is simply a permutation of $\mathbf{C}^T$ for all n_0 due to the periodic extension of $\mathbf{c}[n]$ (3) and because $\mathbf{\Phi}[n] = I$. Thus $\mathbf{O}[n_0]$ has maximal rank because $\mathbf{C}$ (10) is invertible. This satisfies the premise of the applied theorem, which means that the extended conceptual signal model is totally observable. The existence of a suitable observer can be asserted from this, but the task of finding it remains. In general, the problem of finding a linear time variant observer is solved, but observers of this kind are using the inputs of the signal model. There are ways to allow unobservable inputs [5] but the design process is rather complex, not to mention the loss of advantageous numerical properties. Thus the remainder of the article will investigate the observer illustrated on Fig. 2.

C. Impulse response

The goal of this section is to show that the observer not only has an impulse response despite it's time variant nature but it is $\delta[n-N]$. This statement can be formulated equivalently as if $y[n] = \delta[n]$, then $\hat{y}[n] = \delta[n-N]$. To show this the first step is to reorganize the state equation (9) of the observer:

$$\begin{aligned} \hat{\mathbf{x}}[n+1] &= \hat{\mathbf{x}}[n] + \mathbf{g}[n](y[n] - \hat{y}[n]) = \\ &= (\mathbf{I} - \mathbf{g}[n]\mathbf{c}[n]^T)\hat{\mathbf{x}}[n] + \mathbf{g}[n]y[n] = \\ &= (\mathbf{I} - \mathbf{G}\mathbf{e}[n]\mathbf{e}^T[n]\mathbf{G}^{-1})\hat{\mathbf{x}}[n] + \mathbf{g}[n]y[n], \end{aligned} \tag{16}$$

where the last equation uses the definition of $\mathbf{c}[n]$, $\mathbf{g}[n]$ (11) and their biorthogonality (12). It is clear that $\forall n \leq 0\ \hat{\mathbf{x}}[n] = \mathbf{0}$ and by induction for all $1 \leq n \leq N\ \hat{\mathbf{x}}[n] = \mathbf{g}[0]$. If $n = 1$

$$\hat{\mathbf{x}}[1] = (\mathbf{I} - \mathbf{g}[0]\mathbf{c}[0]^T)\mathbf{0} + \mathbf{g}[0]1 = \mathbf{g}[0]. \tag{17}$$

Assuming the induction hypothesis for $1 \leq n < N$

$$\begin{aligned} \hat{\mathbf{x}}[n+1] &= (\mathbf{I} - \mathbf{G}\mathbf{e}[n]\mathbf{e}^T[n]\mathbf{G}^{-1})\mathbf{g}[0] + \mathbf{g}[n]0 = \\ &= \mathbf{g}[0] - \mathbf{G}\mathbf{e}[n]\mathbf{e}^T[n]\mathbf{G}^{-1}\mathbf{G}\mathbf{e}[0] = \mathbf{g}[0]. \end{aligned} \tag{18}$$

The first equation uses the hypothesis, the definition of the state equation (9) and $y[n]$ while the second equation uses the definition of $\mathbf{g}[n]$ (11). The final equation uses the fact that in this case $\mathbf{e}^T[n]\mathbf{e}[0] = 0$ always holds due to the choice of n.

And finally when $n = N+1$

$$\hat{\mathbf{x}}[N+1] = \mathbf{g}[0] - \mathbf{G}\mathbf{e}[N]\mathbf{e}^T[N]\mathbf{G}^{-1}\mathbf{G}\mathbf{e}[0] = \mathbf{0} \tag{19}$$

which is true because the signals were extended periodically, thus $\mathbf{e}[N] = \mathbf{e}[0]$. This means that $\forall n > N\ \hat{\mathbf{x}}[n] = \mathbf{0}$.

Until this point we have proven that when the input is $\delta[n]$ then the state variables of the observer are constant with value $\mathbf{g}[0]$ for N timesteps. Consequently the estimated output for $n \leq 0$ and $n > N$ is $\hat{y}[n] = 0$, and for $0 < n < N$ is

$$\hat{y}[n] = \mathbf{c}^T[n]\hat{\mathbf{x}}[n] = \mathbf{e}^T[n]\mathbf{G}^{-1}\mathbf{G}\mathbf{e}[0] = 0 \tag{20}$$

but for $n = N$

$$\hat{y}[N] = \mathbf{c}^T[N]\hat{\mathbf{x}}[N] = \mathbf{e}^T[N]\mathbf{G}^{-1}\mathbf{G}\mathbf{e}[0] = 1 \tag{21}$$

We have proven that $\hat{y}[n] = \delta[n-N]$, if $y[n] = \delta[n]$. But it is necessary to show the time invariance of this response so in the case when $y[n] = \delta[n-k]$ for some $k \in \mathbb{Z}$. The previous reasoning remains valid, the only difference is that the value of $\hat{\mathbf{x}}$ is $\mathbf{g}[\mathrm{mod}\,(k,N)]$ instead of $\mathbf{g}[0]$. As an important consequence it must be noted that the observer is capable of the error free reconstruction of arbitrary well behaved signals.

D. Expression of state estimation error

The task that remains is to investigate the value of the estimated state variables compared to the ones in the conceptual signal model. Based on the observability of the latter it is expected that reconstruction is possible in the sense of Definition 2.

Unfortunately this is not the case and the intuitive reason lies in the way the estimation is carried out. It is important to note that the observer does not incorporate the inputs of the signal model into the estimation. For example when the state variables are perturbed then the $\varepsilon[n]$ reconstruction error on Fig. 2 is nonzero which clearly indicates that the estimation is not exact. But without this error it would be impossible to tune the estimated variables.

The reconstruction error can be expressed in a closed form with the repeated application of the state equation (9):

$$\hat{\mathbf{x}}[n] = \left(\prod_{i=0}^{n-1} \mathbf{\Phi}[i]\right)\hat{\mathbf{x}}[0] + \sum_{j=0}^{n-1}\left(\prod_{i=j+1}^{n-1} \mathbf{\Phi}[i]\right)\mathbf{g}[j]y[j], \tag{22}$$

where $\mathbf{\Phi}[n] = \mathbf{I} - \mathbf{g}[n]\mathbf{c}^T[n]$. In the corner cases when the initial value of a sum (product) counter is larger than it's final value, then the value of the sum (product) is 0 (1) and it is important to point out that the periodic extension of the signals is frequently used. It was proven in [4] that

$$\prod_{i=0}^{N-1} \mathbf{\Phi}[i] = \prod_{i=0}^{N-1} (\mathbf{I} - \mathbf{g}[i]\mathbf{c}^T[i]) = \mathbf{0} \tag{23}$$

and if $i \neq j$ then due to the biorthogonality of the bases

$$\mathbf{g}[i]\mathbf{c}^T[i]\mathbf{g}[j]\mathbf{c}^T[j] = \mathbf{0}. \tag{24}$$

As a consequence, this means that for any $n \geq N$ the estimation is not dependent on the initial value. Furthermore it won't be dependent on $y[j]$ if $j < n-N$. Considering these and the fact that $y[n] = \mathbf{c}^T[n]\mathbf{x}[n]$ the equation (22) can be simplifed when $n \geq N$:

$$\hat{\mathbf{x}}[n] = \sum_{i=n-N}^{n-1} \mathbf{g}[i]\mathbf{c}^T[i]\mathbf{x}[i]. \tag{25}$$

With this result the reconstruction error can be written as:

$$\varepsilon[n] = \mathbf{c}^T[n]\left(\mathbf{x}[n] - \sum_{i=n-N}^{n-1} \mathbf{g}[i]\mathbf{c}^T[i]\mathbf{x}[i]\right). \tag{26}$$

It is straightforward to show based on (23) that the error is 0 if the state variables are constant for at least N timesteps.

V. Conclusion

The paper reviewed a method to create biorthogonal transforms and an observer which is capable of performing said transforms. For the construction of the observer a detailed introduction of a conceptual signal model and it's extension was necessary. The observability of the extended signal model was shown and the properties of the observer were investigated.

In particular, as a novelty, the observer's ability to reconstruct noncoherent signals was proven which to the best knowledge of the authors has not been done before in this general setting. The main area of application of the observer is the recursive implementation of the DFT [6] [7] [8] and in that special case a modified observer structure can be used which is time invariant so the analysis is different.

Lastly the expression of the state estimation error was derived which proved that the extension of the conceptual signal model is a proper generalization which gives back the original model as a special case. These results serve as a theoretical foundation for the research to come.

Appendix

A. Results About Signal Reconstructibility

Lemma 1: If $\mathbf{e}[n]$ is the nth standard basis vector of $\mathbb{C}^N$ then for all $n \in \mathbb{Z}$ when $y[n] \in \mathbb{C}$, there exists $\mathbf{x}[n] \in \mathbb{C}^N$, such that

$$y[n] = \mathbf{e}^T[n]\mathbf{x}[n].$$

Proof: It suffices to show the existence of $\mathbf{x}$ by giving it. Let's choose

$$\mathbf{x}[n] = \mathbf{e}[n]y[n] \tag{27}$$

which clearly satisfies the statement of the lemma because

$$\mathbf{e}^T[n]\mathbf{x}[n] = \mathbf{e}^T[n]\mathbf{e}[n]y[n] = y[n]. \tag{28}$$

■

Theorem 2: Given a basis ($\mathbf{c}[n]$) and reciprocal basis ($\mathbf{g}[n]$) of $\mathbb{C}^N$, then for all n and $y[n] \in \mathbb{C}$, there exists $\mathbf{x}[n] \in \mathbb{C}^N$, such that

$$y[n] = \mathbf{c}^T[n]\mathbf{x}[n].$$

Proof: Based on it's definition

$$\mathbf{c}[n] = \mathbf{C}\mathbf{e}[n] \tag{29}$$

Substituting this into the equation results in

$$y[n] = \mathbf{c}^T[n]\mathbf{x}[n] = \mathbf{e}^T[n]\mathbf{C}^T\mathbf{x}[n] = \mathbf{e}^T[n]\widetilde{\mathbf{x}}[n] \tag{30}$$

where the last equation defines

$$\widetilde{\mathbf{x}}[n] = \mathbf{C}^T\mathbf{x}[n]. \tag{31}$$

This makes it clear that Lemma 1 can be applied, so

$$\mathbf{C}^T\mathbf{x}[n] = \widetilde{\mathbf{x}}[n] = \mathbf{e}[n]y[n]. \tag{32}$$

Multiplying with $\mathbf{G}$ from the left gives

$$\mathbf{x}[n] = \mathbf{G}\mathbf{C}^T\mathbf{x}[n] = \mathbf{G}\mathbf{e}[n]y[n] = \mathbf{g}[n]y[n], \tag{33}$$

where the first equation is true because the basis and reciprocal basis vectors are biorthogonal (12) and the last is just the definition of $\mathbf{g}[n]$ (11). This proves the existence of $\mathbf{x}$. ■

B. Linear Time Varying Discrete-Time Systems

The following definitions are the straightforward generalizations of the time invariant case described in [9, sec. 9.3].

Definition 1: An LTV DT system has the following form:

$$\mathbf{x}[n+1] = \mathbf{\Phi}[n]\mathbf{x}[n] + \mathbf{\Gamma}[n]\mathbf{u}[n], \tag{34}$$

$$\mathbf{y}[n] = \mathbf{C}[n]\mathbf{x}[n], \tag{35}$$

where $\mathbf{x}[n] \in \mathbb{C}^p$, $\mathbf{y}[n] \in \mathbb{C}^q$ and $\mathbf{u}[n] \in \mathbb{C}^r$.

Definition 2: An LTV DT system is completely observable at time n_0 if and only if there is a finite N such that any $\mathbf{x}[n_0]$ state can be determined exactly from the knowledge of the $\mathbf{u}[n]$ inputs and $\mathbf{y}[n]$ outputs at time instants $n_0 \le n \le n_N$.

Definition 3: An LTV DT system is totally observable if and only if it is completely observable for every n_0 and $n_N > n_0$.

Theorem 3: An LTV DT system is completely observable if

$$\operatorname{rank}(\mathbf{O}[n_0]) = p,$$

where

$$\mathbf{O}[n_0] = \begin{pmatrix} \mathbf{C}[n_0] \\ \mathbf{C}[n_0+1]\mathbf{\Phi}[n_0] \\ \mathbf{C}[n_0+2]\mathbf{\Phi}[n_0+1]\mathbf{\Phi}[n_0] \\ \vdots \\ \mathbf{C}[n_0+N]\mathbf{\Phi}[n_0+N-1]\dots\mathbf{\Phi}[n_0+1]\mathbf{\Phi}[n_0] \end{pmatrix}.$$

Proof: An obvious extension of the proof in [9, sec. 9.3]. ■

In order to check the total observability of the system it is necessary to check the previous theorem for any n_0.

References

[1] J. Kovacevic and A. Chebira, "Life Beyond Bases: The Advent of Frames (Part I)," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 86–104, 2007. Available: https://doi.org/10.1109/MSP.2007.4286567

[2] P. Balazs, M. Dörfler, F. Jaillet, N. Holighaus, and G. A. Velasco, "Theory, Implementation and Applications of Nonstationary Gabor Frames," *Journal of Computational and Applied Mathematics*, vol. 236, no. 6, pp. 1481–1496, 2011. Available: https://doi.org/10.1016/j.cam.2011.09.011

[3] B. Ország and L. Sujbert, "Observer-based discrete Gabor transform," in *2022 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2022, pp. 1–6. Available: https://doi.org/10.1109/I2MTC48687.2022.9806451

[4] G. Péceli, "A common structure for recursive discrete transforms," *IEEE Transactions on Circuits and Systems*, vol. 33, pp. 1035–1036, 1986. Available: https://doi.org/10.1109/tcs.1986.1085844

[5] M. Tranninger, S. Zhuk, M. Steinberger, L. M. Fridman, and M. Horn, "Sliding Mode Tangent Space Observer for LTV Systems with Unknown Inputs," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 6760–6765. Available: https://doi.org/10.1109/CDC.2018.8619848

[6] L. Sujbert, G. Simon, and G. Péceli, "An Observer-Based Adaptive Fourier Analysis [Tips & Tricks]," *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 134–143, 2020. Available: https://doi.org/10.1109/MSP.2020.2982167

[7] Z. Kollar, F. Plesznik, and S. Trumpf, "Observer-Based Recursive Sliding Discrete Fourier Transform [Tips & Tricks]," *IEEE Signal Processing Magazine*, vol. 35, no. 6, pp. 100–106, 2018. Available: https://doi.org/10.1109/MSP.2018.2853196

[8] A. Chauhan and K. M. Singh, "Recursive sliding DFT algorithms: A review," *Digital Signal Processing*, vol. 127, p. 103560, 2022. Available: https://doi.org/10.1016/j.dsp.2022.103560

[9] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems: Theory and Design (2nd Ed.).* Prentice-Hall, Inc., 1990.

Analyzing the Discriminative Power of EEG Microstates Over Mental Tasks

Mihály Vetró
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
vetro@mit.bme.hu

Gábor Hullám
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
gabor.hullam@mit.bme.hu

***Abstract*—Microstate analysis of EEG recordings has long been an instrumental tool for studying the temporal dynamics of whole-brain neuronal networks. The characteristics of EEG microstate sequences have been used mostly for diagnostic purposes, including the detection of schizophrenia, epilepsy, Alzheimer's disease, and early dementia. Aside from diagnostics, the use of this methodology has been limited. In this study, we examine the discriminative power of EEG microstates to differentiate between mental tasks, and we assess the generalizing power of microstate representations over different subjects and recording sessions. For this purpose, we inspect both the characteristics of discrete microstate sequences, as well as various features generated from the association of detected microstates with the continuous EEG data. For demonstration purposes, we use two distinct datasets, which contain recordings from multiple subjects, while performing different mental tasks.**

***Index Terms*—EEG microstates, electroencephalography, statistics, machine learning**

I. Introduction

EEG microstate representations provide a tool to analyze the temporal dynamics of whole-brain neuronal networks. Microstate analysis has been shown to be capable to diagnose schizophrenia [1] [2], epilepsy [3], Alzheimer's disease and early dementia [4]. However, the study of differences between the microstate characteristics of thought processes remain limited. In this paper, we theorize that there is a significant difference in whole-brain temporal dynamics between different mental tasks, aside from the obvious differences which are limited to specific functional areas of the brain. Additionally, we examine the microstate representation as a dimension-reduction tool for EEG data classification.

II. Data used

For this study, two separate datasets were used: (1) the InnerSpeech dataset [5], and (2) the SAM40 dataset [6].

The InnerSpeech dataset contains recordings from 10 subjects, with a total of 3 recording sessions per subject. During a session, the subject was presented with a direction (”up”, ”down”, ”right”, ”left”), and had to perform one of three tasks according to the current phase within the recording session. The task was to either say the given direction out loud as regular speech, ”say” the direction silently as inner speech, or imagine that the dot on the screen in front of the subject is moving towards the designated direction (visualized motion). Every recording session contained 100 direction queues in total, of which 20 belonged to the ”loud speech”, 40 to the ”inner speech” and 40 to the ”visualized motion” tasks. In total, the dataset contains 600 ”loud”, 1200 ”inner” and 1200 ”visual” queues. The recordings were made using 128 EEG electrodes arranged in the standard BioSemi128 montage at a sampling frequency of 256 Hz.

The SAM40 dataset contains 40 subjects in total, with 3 recording sessions for every subject. In every recording session, the subject performed 3 mental tasks, each for 25 seconds. All of these tasks were simple problems which required binary decisions, essentially ”yes or no” answers. The first task in every session was the ”stroop color-word test”, in which the subject had to decide if the word on the screen matched the color with which it was written (e.g. if the word ”green” was typed with green letters, or a different color). The next task was the ”mirror image symmetry test”, in which the task of the subject was to determine if the two images on the screen (consisting of large colored tiles) were symmetrical or not. Lastly, the final task for every session was the ”arithmetic test”, where a simple equation was presented on the screen, and the subject had to determine if it was correct or not. Every one of these tasks were performed multiple times in every recording session, where upon completion, the subject was presented with a new task within the same category, during the time interval (25 seconds) designated for the given task. In total, the dataset contains 120 ”stroop”, 120 ”symmetry” and 120 ”arithmetic” recordings, each 25 seconds long. The recordings were made using 32 EEG electrodes in the standard 1020 montage, at a sampling frequency of 128 Hz.

III. Methods

To determine the discriminative power of microstates over the different mental tasks described in section II, several statistical and machine learning methods were applied, including cluster ”goodness of fit” metrics, analysis of statistical significance, Support Vector Machines (SVM), Markov-chain classifiers, and Convolutional neural networks. The methods used for classification were evaluated using both randomized and subject-wise cross-validation, to assess cross-subject generalization capabilities. The preprocessing and analysis was performed using the MNE [7] and Pycrostates [8] libraries.

A. *Preprocessing and data standardization*

For both datasets, standard EEG preprocessing techniques were applied: the powerline noise was removed using a notch-filter at 50Hz, slow drifts and high-frequency noise were countered using a band-pass filter between 1 Hz and 40 Hz. Afterwards, automatic artifact removal was performed for the InnerSpeech dataset using Independent Component Analysis (ICA) [9], where the correlation with the dedicated electrooculographic (EOG) electrodes (placed directly above and below the right eye, as well as on both temples) was used as an exclusion criteria to identify the components containing blink and eye movement artifacts. Note that in the case of the SAM40 dataset, artifact removal was previously carried out by Gosh et al. [6]. After preprocessing, session-wise standardization was performed by calculating the Z-score of the resulting voltage levels over every session, therefore bringing the expected value and standard deviation of the sessions into a common domain.

B. *Microstate clustering*

Microstates are quazi-stable, transient states that can be observed in an EEG recording [10]. In order to determine the microstates that will be used for further analysis, the local maxima of the Global Field Power (GFP) was determined using a simple peak detection algorithm, which relies on neighbouring values. As previous research has showed, a stable microstate usually persists for around 100 ms, except for GFP fluctuations and changes in polarity [10]. From this, the minimum peak distance of the peak detection algorithm was set to 50 ms, therefore drawing roughly between 1 and 3 samples from every persistent microstate. After peak detection, the identified GFP peaks were sub-sampled so every session (from every subject) is represented evenly during clustering.

From the evenly sampled set of GFP peaks, the set of microstates was determined using a Modified K-Means algorithm [11], with 100 independent initializations, where every initialization was run until convergence. To determine the optimal number of clusters, the Calinski-Harabasz score [13] was used. Every possible number of clusters were considered within an inclusive range from 2 to 8. The inverse of the Pearson product-moment correlation coefficient was used as a distance function both for clustering and cluster evaluation.

Finally, the EEG data was segmented into the microstates thus defined, so as every time-point is categorized into the microstate that it has the maximum absolute correlation with. This approach deliberately disregards polarity, since polarity inversions can occur during the persistence of an otherwise stable microstate. Additionally, GFP fluctuations are also disregarded, because a change in GFP should not affect the relative distances of the different microstates from the momentary topographical brain state.

C. *Microstate characteristics*

To obtain overall characteristics from the microstate sequences belonging to the epochs (representing different mental tasks), a set of length-independent metrics were utilized,

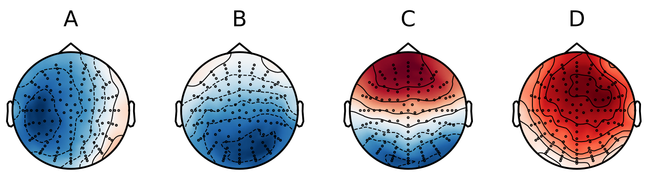


Fig. 1. Detected microstates of the InnerSpeech dataset, named using the standard notation.

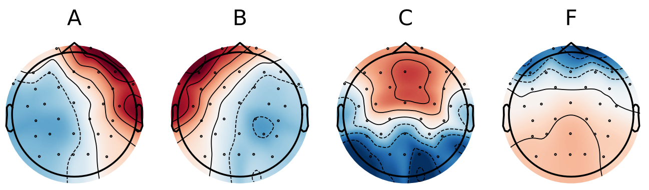


Fig. 2. Detected microstates of the SAM40 dataset, named using the standard notation.

adapted from diagnostic studies [1] [2]. The following metrics were computed independently for every microstate:

- Mean Correlation: the mean correlation value with the given microstate, taken from the timepoints that have been categorized to this microstate.
- Global Explained Variance (GEV) [12]: the sum of GEV values of every timepoint that has been categorized to the given microstate.
- Time Coverage: the ratio of total time covered by the given microstate.
- Mean Duration: the average time interval (in seconds) for which the given microstate persists.
- Occurrence: how many times the given microstate occurs per second (Hz).

The use of such metrics is particularly beneficial for the SAM40 dataset, where stimuli occur multiple times within the 25 second cue interval with an irregular rate (depending on how many times a subject can perform the actual task within the allowed time-frame), therefore rendering the approach of classic event-related potential (ERP) analysis impractical.

To determine if there is any significant difference in the calculated microstate characteristics between different mental tasks, a statistical test was performed. For this, the difference between the means (expected values) of every metric was taken for every class (mental task), and then randomized values were taken by perturbing the class labels and recalculating the differences between the expected values for the newly assigned (randomized) classes. In this case, the null-hypothesis assumes that the difference between the examined classes (regarding the examined characteristic) was the result of random chance, instead of a real underlying phenomenon. The probability (p) of the null hypothesis was estimated given the original difference between the expected values, assuming a normal distribution over the randomized difference values. The difference was considered significant, if $p < 0.01$.

D. Discriminative power of microstate sequences

To assert the discriminative power of microstate sequences, multiple approaches were considered.

First, the discriminative power of the microstate characteristics was analyzed, based on the microstate-specific parameters described in section III-C. For this purpose, Support Vector Machines (SVM) were utilized, using a Radial Basis Function (RBF) kernel. Also, the discriminative power of feed-forward neural networks were investigated over the same characteristics, using a network with 3 hidden layers, having 64, 32 and 16 units respectively, and ReLU as the activation function of the hidden layers.

Next, the discrete microstate sequences were investigated, using a Markov-Chain classifier. This classifier fits a Markov-Chain model to the data from each class, therefore computing the expected state-transition probabilities within the sequences belonging to the different classes. During inference, the log-probability of the input sequence is computed from the Markov-Chain model for every class, and the prediction of the classifier is the class for which its associated Markov-Chain model returned the highest log-probability.

Finally, the continuous microstate correlations were considered as a type of dimension-reduction method for EEG data, for the purpose of classification. In this case, the correlation value was computed for every time-point in the EEG (in the form of the dot-product of the momentary EEG state and the expected electrode values of the microstate), which resulted in a unique time-series for every considered microstate. Also, the Global Field Power was attached to the input as an additional possible indicator. As a result, the original EEG data is represented by $k + 1$ total "channels", where k is the number of considered microstates, and the number of time-points is unchanged. For the classification for these correlation sequences, the EEGNet [14] neural network architecture was utilized, with a temporal kernel size of $f_s/2$ (according to the recommendation by Lee et. al. [15], where f_s is the sampling frequency of the EEG recording), and spatial dropout was used. For reference, the classification performance of the EEGNet architecture was also investigated (using the same $f_s/2$ temporal kernel length [15]) on the original, standardized EEG data, using the Z-score standardization, as mentioned in section III-A.

IV. Results

For both the InnerSpeech and the SAM40 datasets, $k = 4$ was chosen based on the Calinski-Harabasz score, which was maximal for the cluster count of four in both datasets. The resulting microstates for the InnerSpeech dataset can be seen in figure 1, and for the SAM40 dataset in figure 2. The microstate labels were determined based on the notation summarized by Michael et. al. [16]. Although the microstates were categorized into the closest reference notation, there are noticable differences between the topographies of the same microstates from the two datasets. These differences can be largely attributed to the specific properties of the (relatively small) patient groups and the measuring equipment.

TABLE I
Probabilities of the null-hypothesis for the InnerSpeech dataset

Loud speech

MS	GEV	Correlation	Duration	Occurrence	Time cov.
A	**<0.0001**	**<0.0001**	0.2672	0.8832	0.2231
B	**0.0042**	0.0535	0.7859	0.0464	0.298
C	**<0.0001**	0.3176	**<0.0001**	**<0.0001**	**<0.0001**
D	**0.0003**	**0.0002**	0.2063	0.0599	0.0126

Inner speech

MS	GEV	Correlation	Duration	Occurrence	Time cov.
A	0.0116	**0.0049**	0.6837	0.3777	0.798
B	0.1184	0.0171	0.6272	0.8764	0.7447
C	**0.0037**	0.0607	0.0344	0.3245	0.0335
D	0.1301	0.4594	0.8463	0.0447	0.2393

Visualized motion

MS	GEV	Correlation	Duration	Occurrence	Time cov.
A	0.0856	**0.0035**	0.5863	0.3308	0.3949
B	0.445	0.7902	0.4191	0.0675	0.5613
C	0.1151	0.6078	**0.0001**	0.0126	**0.0002**
D	0.1398	0.0115	0.1292	0.8856	0.3998

TABLE II
Probabilities of the null-hypotheses for the SAM40 dataset

Stroop color word task

MS	GEV	Correlation	Duration	Occurrence	Time cov.
A	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**
B	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**
C	**<0.0001**	0.3057	**<0.0001**	**<0.0001**	**<0.0001**
F	0.5306	0.0134	0.3087	0.806	0.7126

Mirror image task

MS	GEV	Correlation	Duration	Occurrence	Time cov.
A	0.7686	0.5462	0.6188	0.2551	0.791
B	**0.0001**	**0.0037**	0.0764	**0.0095**	**0.0044**
C	**<0.0001**	0.5511	**<0.0001**	**0.0079**	**<0.0001**
F	0.1548	0.4334	0.5216	**0.0081**	0.1636

Arithmetic task

MS	GEV	Correlation	Duration	Occurrence	Time cov.
A	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**
B	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**
C	**<0.0001**	0.7552	**<0.0001**	**<0.0001**	**<0.0001**
F	0.5972	0.2266	0.7814	0.0257	0.421

A. Statistical significance of characteristic differences

Afterwards, the microstate characteristics were computed according to section III-C, and the probability (p) for the null hypothesis was computed for every characteristic and every mental task, in a one-versus-rest manner. The such-acquired null-hypothesis probabilities for the InnerSpeech dataset can be seen in table I, and for the SAM40 dataset in table II. In both tables, the null-hypothesis probabilities of the characteristics which satisfied the significance criteria ($p < 0.01$) are highlighted. For the InnerSpeech dataset, the characteristics of the loud speech task showed the most significant difference from the rest, suggesting that it is the easiest to discriminate by the examination of whole-brain temporal dynamics. The other two tasks showed less significant difference with regards to the examined characteristics, which suggests that either the semantic difference is small between inner speech and visualized motion, or it cannot be expressed using the characteristics of microstate representations. In the case of the SAM40 dataset, all three mental tasks showed significant difference with regards to multiple microstates and associated characteristics, with F being the least indicative microstate.

TABLE III
CLASSIFICATION ACCURACY OF THE APPLIED METHODS

	Data type	Characteristics		Discrete sequence	Correlation	EEG data
	Model	SVM	DNN	MCC	EEGNet	EEGNet
Dataset	**CV-method**					
InnerSpeech	random	40%*	40%*	40%*	47% (±1%)	58% (±2%)
	subject	40%*	40%*	40%*	46% (±3%)	51% (±4%)
SAM40	random	53% (±7%)	62% (±8%)	57% (±4%)	63% (±7%)	74% (±8%)
	subject	53% (±7%)	60% (±4%)	54% (±5%)	62% (±9%)	71% (±10%)

*Chance-level predictions

B. *Classification based on microstate sequences*

To assess the discriminative power of microstates, multiple approaches and representations were examined, as it is described in section III-D. The objective for both datasets was to discriminate between the three mental tasks that were conducted by the subjects during the EEG recording. It is important to note, that the three classes of the InnerSpeech dataset are distributed in a 20-40-40 manner (with pronounced speech being the least-represented class), therefore the chance level for accuracy regarding that dataset is 40%. In contrast, the three examined classes in the SAM40 dataset are evenly distributed, therefore the chance accuracy level for that dataset is 33.3%. Additionally, for all classification methods, two different cross-validation methods were applied: (1) randomized cross-validation, where all the samples were randomly assigned to either the training or the test dataset, and (2) subject-wise cross-validation, where the training and the test datasets were divided on a subject level.

The classification accuracy of the applied methods over both datasets and cross-validation techniques is shown in table III. For the InnerSpeech dataset, the classifications based on the microstate characteristics and discrete microstate sequences produced chance-level accuracies, with the only successful attempts being the classifications based on microstate correlations and the original EEG data (the latter of which is provided for reference). For the SAM40 dataset, the classification based on a deep feed-forward neural network produced a similar level of accuracy as the EEGNet architecture over the microstate correlation data. However, in the case of both datasets, the highest level of accuracy was achieved by the EEGNet architecture applied over the original EEG data, without the utilization of microstate representations.

V. DISCUSSION

Based on the results of the statistical analysis conducted in this study, significant differences can be observed in the whole-brain temporal dynamics while performing different mental tasks. Also, the two analyzed datasets produced drastically different results both in terms of classification accuracy (shown in table III) and statistical difference over the microstate characteristics (shown in tables I and II). We theorize that this difference between the datasets can be attributed to the different nature of the performed tasks. In the SAM40 dataset, the subjects had to solve actual problems, which required complex thought processes (producing higher brain activity in the process), while the InnerSpeech dataset was recorded on subjects performing less demanding non-interactive tasks. Finally, the results of the classification attempts suggest that although the microstate representations produce input data of significantly lower dimensionality, some of the essential, task-specific information is lost during conversion. Additionally, the inter-subject generalization capabilities of microstate representations are demonstrated by the negligible differences between the classification accuracies of the randomly cross-validated and subject-wise cross-validated attempts.

REFERENCES

[1] M. Baradits, I. Bitter, P. Czobor, ”Multivariate patterns of EEG microstate parameters and their role in the discrimination of patients with schizophrenia from healthy controls”, vol. 288, pp. 112938, Psychiatry Res., 2020.

[2] A. Keihani, S. S. Sajadi, M. Hasani, F. Ferrarelli, ”Bayesian Optimization of Machine Learning Classification of Resting-State EEG Microstates in Schizophrenia”, Brain Sci.,vol. 12, no. 11, pp. 1497, 2022.

[3] J. B. Kill, P. M. Ciarelli, K. F. Côco, ”Analysis of EEG microstates to predict epileptic seizures in an online approach”, Res. Biomed, vol. 38, pp. 409–421, 2022.

[4] L. Tait, F. Tamagnini, G. Stothart et al, ”EEG microstate complexity for aiding early diagnosis of Alzheimer's disease”, Sci. Rep., vol. 10, pp. 17627, 2020.

[5] N. Nieto, V. Peterson et al., ”Thinking out loud, an open-access EEG-based BCI dataset for inner speech recognition while performing Stroop color-word test, arithmetic task, and mirror image recognition task”, Sci. Data, vol. 9, no. 52, 2022.

[6] R. Ghosh, N. Deb et. al., ”SAM 40: Dataset of 40 subject EEG recordings to monitor the induced-stress”, Data in Brief, vol. 40, pp. 107772, 2022.

[7] A. Gramfort, M. Luessi, E. Larson et. al., ”MNE software for processing MEG and EEG data”, Neuroimage, vol. 86, pp. 446-460, 2014.

[8] Férat et al., ”Pycrostates: a Python library to study EEG microstates”, Journal of Open Source Software, vol. 7, no. 78, pp. 4564, 2022.

[9] P. Comon, ”Independent component analysis, A new concept?”, Signal Processing, vol. 36, no. 3, pp. 287-314, 1994.

[10] D. Van de Ville, J. Britz, C. M. Michel, ”EEG microstate sequences in healthy humans at rest reveal scale-free dynamics”, PNAS, vol. 107, no. 42, pp. 18179-18184, 2010.

[11] R. D. Pascual-Marqui, C. M. Michel, D. Lehmann, ”Segmentation of brain electrical activity into microstates: model estimation and validation”, IEEE Trans Biomed Eng., vol. 42, no. 7, pp. 658-65, 1995 Jul.

[12] M. M. Murray, D. Brunet, C. M. Michel, ”Topographic ERP Analyses: A Step-by-Step Tutorial Review”, Brain Topography, vol. 20, no. 4, pp. 249-64, 2008 Jun.

[13] T. Caliński, J. Harabasz, ”A dendrite method for cluster analysis”, Communications in Statistics, vol. 3, no. 1, pp. 1-27. 1974.

[14] V. J. Lawhern, A. J. Solon, N. R. Waytowich et. al., ”EEGNet: a compact convolutional neural network for EEG-based brain-computer interfaces”, Journal of Neural Engineering, vol. 15, no. 5, pp. 056013, 2018.

[15] Y. E. Lee, S. H. Lee, ”EEG-Transformer: Self-attention from Transformer Architecture for Decoding EEG of Imagined Speech”, 10th Int. Winter Conf. on Brain-Computer Interface (BCI), pp. 1-4, 2021.

[16] C. M. Michel, T. Koenig, ”EEG microstates as a tool for studying the temporal dynamics of whole-brain neuronal networks: A review”, NeuroImage, vol. 180, pp. 577-593, 2018.

Thoracic Spine Segmentation Based on CT Images

Gábor Révy, Dániel Hadházi, Gábor Hullám
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
{revy, hadhazi, hullam.gabor}@mit.bme.hu

***Abstract*—Automatic vertebrae localization and segmentation in computed tomography (CT) are fundamental for computer-aided detection (CADe) and computer-aided diagnosis (CADx) systems. However, they remain challenging due to the high variation in spinal anatomy among patients. In this paper, we propose a simple, model-free approach for automatic CT vertebrae localization and segmentation. The segmentation pipeline consists of 3 stages. In the first stage the center line of the spinal cord is estimated using convolution. In the second stage a baseline segmentation of the spine is created using morphological reconstruction and other classical image processing algorithms. Finally, the baseline spine segmentation is refined by limiting its boundaries using simple heuristics based on expert knowledge. We evaluated our method on the COVID-19 subdataset of the CTSpine1K dataset. Our solution achieved a dice coefficient of 0.8160±0.0432 (mean±std) and an intersection over union of 0.6914±0.0618 for spine segmentation. The experimental results have demonstrated the feasibility of the proposed method in a real environment.**

***Index Terms*—spine segmentation, CT, image processing, expert system**

I. Introduction

The segmentation of the rib cage can be an important stage in computer-aided detection (CADe) and computer-aided diagnosis (CADx) systems. In some cases, the spine, as part of the rib cage, may even need to be segmented separately. Current solutions for dealing with this task generally use explicit models of some kind. However, to create sufficiently robust models, large amounts of well-labelled heterogeneous data are required. This is not always available. Another drawback of these solutions is that they focus on segmenting the vertebrae individually, thus require a manually defined bounding box, which makes these algorithms only semi-automatic. Furthermore, the individual labeling is not necessary for all applications. In this work, we propose a fully-automatic, explicit model-free algorithm for spine segmentation that utilizes medical expert knowledge.

II. Related work

The segmentation of the spine is a currently active area of research. These solutions usually utilize a model to create an accurate segmentation. The two main approaches are the neural network and the statistical model-based solutions. There are several studies [1]–[4] investigating the use of U-Net [5] for spine segmentation. Cheng *et al.* [6] designed a two-stage Dense-U-Net for automatic CT vertebrae localization and segmentation. In the first step the centroid of the vertebrae was localized on the 2D slices using a 2D-Dense-U-Net. Based on the centroids, a 3D-Dense-U-Net is utilized to segment the vertebrae within the region-of-interests that are detected during the first stage. Vania *et al.* [7] combined a convolutional neural network (CNN) and a fully convolutional network (FCN). They also utilize class redundancy to improve the segmentation results. This means that the area around the spine is masked with a different value in the training data, forcing the model to accurately distinguish between the spine and its surroundings.

To determine a statistical shape model (SSM), Dryden *et al.* [8] utilized labeled landmark points of the examined shape (body). This model is fitted to the detected landmark points in case of the actual problem by registration. Benameur *et al.* [9] use a 2.5D statistical registration model from biplanar radiographic images for the 3D reconstruction of the vertebrae of a scoliotic spine.

Khandelwal *et al.* [10] designed a two-step pipeline for vertebrae segmentation utilizing the active shape model algorithm [11], which is a type of statistical shape analysis. In the first step of their pipeline the entire spine is segmented as a single surface object using region-based geometric flows. This step requires a seed point identified by the user. In the second step, the geometry of the vertebrae is taken into account during the segmentation of the individual vertebrae. They utilize a shape prior that is fitted based on the density values of the surrounding area of the spine.

III. Method

The segmentation consists of 3 main steps. First, the center of the spinal cord is estimated on each axial slice. Then, morphological reconstruction is applied to create a baseline segmentation of the spine. Finally, this segmentation is refined to obtain the final segmentation.

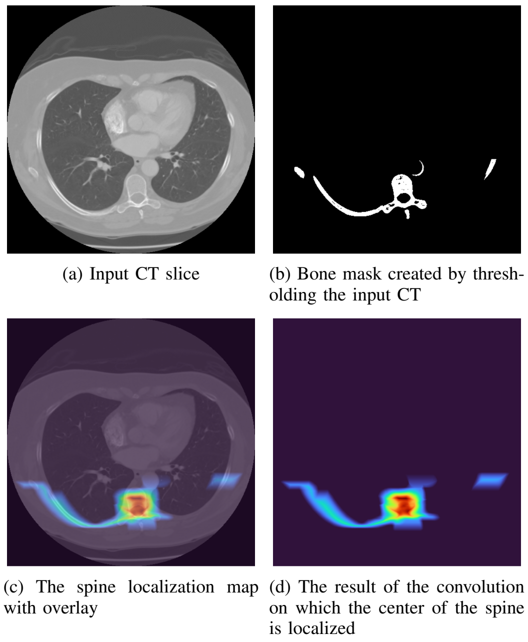
A. Spinal cord center estimation

To estimate the center of the spinal cord in the axial slices, first, the Hounsfield unit (HU) values (see Figure 1a) are thresholded to highlight the values above bone density. We set this threshold to 150 HU to ensure that the bone values are included in the segmentation even in the presence of noise. You can see the resulting segmentation mask in Figure 1b.

This research was funded by the National Research, Development, and Innovation Fund of Hungary under Grant TKP2021-EGA-02.

Although with this solution, the thresholding is robust, if contrast material is present, the heart and other blood vessels are also highlighted. To address this issue, only the posterior half of the body is taken into account on the axial slices. Another problem caused by the presence of the contrast agent concerns the descending aorta. The descending aorta runs close to the spine thus its segmentation might merge with the segmentation of the spine. This problem is solved by removing the segmentation of the descending aorta from the resulting mask. To create the segmentation of the descending aorta a Hough transform-based method [12] is utilized.

The center of the spinal cord is localized using convolution. The mask created by thresholding is convolved with a disk kernel on each axial slice. The diameter of the kernel is set to match the size of an average vertebra. The result of the convolution step is shown in Figure 1d.


(a) Input CT slice

(b) Bone mask created by thresholding the input CT

(c) The spine localization map with overlay

(d) The result of the convolution on which the center of the spine is localized

Fig. 1: The steps of estimating the center of the spinal cord. LIDC_0018 [13]

To make the localization more robust the sagittal slice that contains the largest segmented area is selected. This is expected to select the sagittal slice at the center of the spine. This sagittal slice defines a column in the axial slices, with a band around it in which we look for the center. In this band, on each slice, the point with the largest value obtained from the convolution is selected. The resulting points are smoothed by polynomial fitting using the RANSAC [14] method with L1 norm fidelity term, which makes it robust to outliers. The resulting cord center is shown in Figure 2.

B. Baseline spine segmentation

The segmentation created by thresholding the density values and the resulting estimated spinal cord center points are utilized to create a baseline spine segmentation. First, rays

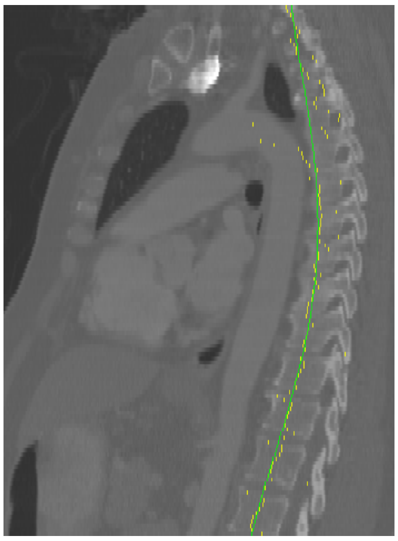


Fig. 2: The estimated spinal cord center points (yellow) and the fitted polynomial (green) projected onto the center sagittal slice determined by the algorithm. LIDC_0018 [13]

are cast from the center points to the anterior and posterior directions. The points are recorded where the rays intersect the segmentation. Points far from the center are discarded. The remaining points serve as seed points for morphological reconstruction [15], which is also based on the mask produced by thresholding (bone mask). The morphological reconstruction is performed on each slice separately. At this point, the voxels belonging to the spinous process might not be included in the reconstructed mask, since they might belong to a component other than the seed points of the reconstruction on the bone mask. This can be solved by dilating the reconstructed mask across the slices, masking it with the bone mask (using binary *AND* operation) and applying another morphological reconstruction. This solution had two drawbacks. The first problem is that the segmentation of the descending aorta is not perfect and the edge of the descending aorta might be included in the bone mask. Therefore, the dilation across the slices combined with the second reconstruction may result in a segmentation that flows into the unsegmented part of the descending aorta and into the posterior part of the heart. This is solved by applying the dilation across the slices only in the inferior direction (i.e. away from the head). This means that the segmentation of a slice is affected only by the segmentation of the slices in superior direction (i.e. towards the head). This way the imprecise segmentation of the descending aorta cannot flow into the superior slices. The other problem is similar: part of the descending aorta and other small vessels next to the vertebral body may be included in the bone mask. This can lead to the inclusion of unnecessary objects in the segmentation after the second morphological reconstruction is applied. To overcome this phenomenon the iteration number of the morphological reconstruction is limited in the anterior direction from the center point. An example of the resulting

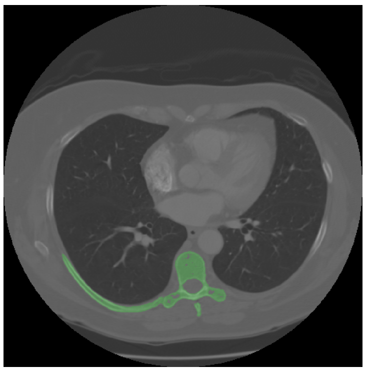


Fig. 3: The resulting baseline spine segmentation. This segmentation is refined by subsequent steps to reduce the segmentation of the ribs. LIDC_0018 [13]

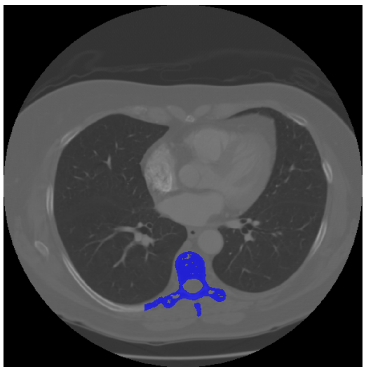


Fig. 4: Example for a resulting spine segmentation. LIDC_0018 [13]

segmentation can be seen in Figure 3.

C. Limiting the boundaries of the segmentation

As it can be seen in Figure 3, the segmentation resulting in the previous step includes unnecessary parts: typically the ribs are included in the segmentation. The performed steps described in this section aim to reduce this type of error. These algorithms utilize expert medical knowledge to determine bounding lines for the segmentation mask on the axial slices.

We start with the reconstructed mask and the spinal cord center from the previous step to create these lines. In the axial slices, this reconstructed mask is used to determine the top of the vertebral body. Several columns of this mask are selected in a band around the estimated center of the spinal cord. The upper part of the segmentation in this band is considered the most anterior part of the vertebral body.

Maximum intensity projection is performed perpendicular to the sagittal slices, followed by thresholding to determine the end of the spinous process. This can be obtained from the segmentation mask directly. Since the end of the spinous process can be found further back than the ribs, the most posterior point of the resulting segmentation is considered to be the end of the spinous process.

In the next step, the cumulative sum of the number of segmentation pixels across the columns in each axial slice is first calculated, then the median of the cumulative sum is used to calculate a more accurate center column of the vertebral body.

The upper third of the vertebral body is used to measure the width of the vertebral body in each slice, which can be derived from the top of the vertebral body and the end of the spinous process. The calculated width of the vertebral body allows us to estimate the maximal width of the whole vertebra. To precisely separate the ribs from the vertebrae a model would be required. Here, we utilize a simple heuristics that ensures that the segmentation includes the transversal process and cuts off a significant part of the segmentation of the ribs. The edges of the vertebral body are moved by its width in both directions. This way two lines are defined from which the segmentation is discarded outwards. Figure 4 shows an example of the resulting segmentation.

D. Evaluation

To evaluate the performance of the proposed segmentation algorithm the COVID-19 [16], [17] subdataset of the CTSpine1K [18] dataset was used. The CT scans can be retrieved from TCIA [19]. The COVID-19 dataset consists of chest CT scans of full thoracic view from 632 patients with COVID-19 infections. 20 of these were selected and annotated in the CTSpine1K dataset. To select CT scans with full thoracic view for evaluation was crucial, since the segmentation pipeline relies on certain anatomical features. Furthermore, the CT scans with reduced field of view are usually obtained to segment the vertebrae individually and in such cases the localisation of the spine is not required.

Two metrics were used to indicate the performance of the algorithm:

- Intersection over union (IoU) is the ratio of the intersection and union of the segmentation mask defined by the algorithm (A) and the manual labeling (B):

$$\mathrm{IoU}(A,B)=\frac{|A\cap B|}{|A\cup B|}=\frac{|A\cap B|}{|A|+|B|-|A\cap B|}$$

- The Dice similarity coefficient (DSC) equals twice the intersection of the segmentation mask volumes divided by the sum of the volumes:

$$\mathrm{DSC}(A,B)=\frac{2|A\cap B|}{|A|+|B|}=\frac{2|A\cap B|}{|A\cup B|+|A\cap B|}$$

Table I shows the results of the evaluation.

	IoU	DSC
avg	0.6914	0.8160
std	0.0618	0.0432
min	0.5654	0.7223

TABLE I: Evaluation results of the proposed segmentation algorithm.

Table II shows the evaluation results compared to methods introduced in the related work section. The comparison shows that the accuracy of the presented method is at the bottom of the range. However, it is worth noting, that [3] used only 9 scans for training and 1 scan for evaluating their neural network. Furthermore, [1] and [2] evaluated their model on the training dataset. In summary, most of these models achieve a high accuracy based on a relatively small dataset of samples with similar characteristics. However, their application to datasets with different properties may require additional steps, such as the retraining and fine-tuning of models. In contrast, our method is more robust, it does not require a labeled training dataset, and thus no retraining step is needed.

paper	IoU	DSC	method
[1]	0.7228	0.8477	U-Net + ASPP
[2]	0.9433	0.9708	3D Dense-U-Net Network
[3]	0.9193	0.9580	RAR-U-Net
[6]	0.911	0.953±0.014	two-stage Dense-U-Net
[7]	0.9336	0.9428	CNN
[10]	-	0.9236	Geometric Flows + Shape Priors
ours	0.6914	0.8160±0.0432	classical image processing

TABLE II: Comparison of results with related algorithms.

IV. Conclusion

In this paper, we have proposed an explicit model-free segmentation technique for spinal segmentation. The segmentation system uses a CT scan of full thoracic field of view as its input. Classical image processing algorithms that exploit medical expertise were utilized. Our approach has the advantage of not requiring an explicit model and it is fully automatic. This method can be applied when a fast and simple segmentation is preferred and it is not critical that the ribs do not completely separate from the spine on the segmentation. In such cases, however, when individual vertebral segmentation is required, a model-based approach is necessary.

Acknowledgement

The authors would like to thank the Multi-National NIH Consortium for CT AI in COVID-19. The authors acknowledge the National Cancer Institute and the Foundation for the National Institutes of Health, and their critical role in the creation of the free publicly available [20] LIDC/IDRI Database [13] used in this study. This research was funded by the National Research, Development, and Innovation Fund of Hungary under Grant TKP2021-EGA-02.

References

[1] Z. Yang, L. Chen, T. Fu, Z. Yin, and F. Yang, "Spine image segmentation based on u-net and atrous spatial pyramid pooling," in *Journal of Physics: Conference Series*, vol. 2209, no. 1. IOP Publishing, 2022, p. 012020.

[2] M. Kolařík, R. Burget, V. Uher, K. Říha, and M. K. Dutta, "Optimized high resolution 3d dense-u-net network for brain and spine segmentation," *Applied Sciences*, vol. 9, no. 3, p. 404, 2019.

[3] Z. Wang, Z. Zhang, and I. Voiculescu, "Rar-u-net: a residual encoder to attention decoder by residual connections framework for spine segmentation under noisy labels," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 21–25.

[4] C. Buerger, J. von Berg, A. Franz, T. Klinder, C. Lorenz, and M. Lenga, "Combining deep learning and model-based segmentation for labeled spine ct segmentation," in *Medical Imaging 2020: Image Processing*, vol. 11313. SPIE, 2020, pp. 307–314.

[5] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[6] P. Cheng, Y. Yang, H. Yu, and Y. He, "Automatic vertebrae localization and segmentation in ct with a two-stage dense-u-net," *Scientific Reports*, vol. 11, no. 1, pp. 1–13, 2021.

[7] M. Vania, D. Mureja, and D. Lee, "Automatic spine segmentation from ct images using convolutional neural network via redundant generation of class labels," *Journal of Computational Design and Engineering*, vol. 6, no. 2, pp. 224–232, 2019.

[8] I. L. Dryden and K. V. Mardia, *Statistical shape analysis: with applications in R*. John Wiley & Sons, 2016, vol. 995.

[9] S. Benameur, M. Mignotte, S. Parent, H. Labelle, W. Skalli, and J. de Guise, "3d/2d registration and segmentation of scoliotic vertebrae using statistical models," *Computerized Medical Imaging and Graphics*, vol. 27, no. 5, pp. 321–337, 2003.

[10] P. Khandelwal, D. L. Collins, and K. Siddiqi, "Spine and individual vertebrae segmentation in computed tomography images using geometric flows and shape priors," *Frontiers in Computer Science*, p. 66, 2021.

[11] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models-their training and application," *Computer vision and image understanding*, vol. 61, no. 1, pp. 38–59, 1995.

[12] U. Kurkure, O. C. Avila-Montes, and I. A. Kakadiaris, "Automated segmentation of thoracic aorta in non-contrast ct images," in *2008 5th IEEE Int. Symp. on Biomed. Imaging*. IEEE, 2008, pp. 29–32.

[13] Armato III, Samuel G., G. McLennan, L. Bidaut *et al.*, "Data from lidc-idri," 2015. [Online]. Available: https://wiki.cancerimagingarchive.net/x/rgAe

[14] H. Cantzler, "Random sample consensus (ransac)," *Institute for Perception, Action and Behaviour, Division of Informatics, University of Edinburgh*, vol. 3, 1981.

[15] K. Robinson and P. F. Whelan, "Efficient morphological reconstruction: a downhill filter," *Pattern Recognition Letters*, vol. 25, no. 15, pp. 1759–1767, 2004.

[16] P. An, S. Xu, S. A. Harmon, E. B. Turkbey *et al.*, "Ct images in covid-19," 2020. [Online]. Available: https://wiki.cancerimagingarchive.net/x/o5QvB

[17] S. A. Harmon *et al.*, "Artificial intelligence for the detection of covid-19 pneumonia on chest ct using multinational datasets," *Nature communications*, vol. 11, no. 1, pp. 1–7, 2020.

[18] Y. Deng, C. Wang, Y. Hui, Q. Li, J. Li, S. Luo, M. Sun, Q. Quan, S. Yang, Y. Hao *et al.*, "Ctspine1k: A large-scale dataset for spinal vertebrae segmentation in computed tomography," *arXiv e-prints*, pp. arXiv–2105, 2021.

[19] K. Clark, B. Vendt, K. Smith, J. Freymann, J. Kirby, P. Koppel, S. Moore, S. Phillips, D. Maffitt, M. Pringle *et al.*, "The cancer imaging archive (tcia): maintaining and operating a public information repository," *Journal of digital imaging*, vol. 26, no. 6, pp. 1045–1057, 2013.

[20] K. Clark, B. Vendt *et al.*, "The cancer imaging archive (TCIA): Maintaining and operating a public information repository," *Journal of Digital Imaging*, vol. 26, no. 6, pp. 1045–1057, Jul. 2013. [Online]. Available: https://doi.org/10.1007/s10278-013-9622-7

Handling Uncertainty in Error Propagation Analysis

András Földvári
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
andras.foldvari@edu.bme.hu

András Pataricza
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
pataricza.andras@vik.bme.hu

***Abstract*—Error propagation analysis (EPA) is a systematic model-based approach to assess the impact of incidental or malicious faults in the dependability and security analysis of complex systems.**

Its main purpose is to estimate the most severe failures in the system under evaluation. It can be extended to evaluate the efficiency of built-in error protection and mitigation mechanisms.

However, during the EPA, uncertainties may arise, which may affect the outcome - this way, the validity of the analysis - and lead to escaping faults. Uncertainties can originate from two primary sources. Firstly, epistemic-type uncertainties express that there may be parts of the analyzed system that are unknown to the domain expert. Secondly, aleatory uncertainties may arise from incorrect or incomplete modeling of the system or even from the non-deterministic operation (physical processes). Our approach extends the known EPA models by handling the uncertainties via rough set theory, an advanced mathematical paradigm to generate approximate descriptions of the system behavior.

***Index Terms*—error propagation analysis, rough set theory, uncertainty**

I. Introduction

Compliance with data security and dependability requirements is essential for the design and operation of critical systems. The purpose of such evaluations is a systematic assessment of whether the propagation of hypothetical failures does not cause critical failures in the services delivered by the target system.

The most thoroughgoing impact analysis method is Error Propagation Analysis (EPA) [1], [2], which takes a model of the error-free system, generates a set of mutations from the anticipated fault set, and evaluates these combinations over the hypothetical input sequences. In this way, the EPA analyzes the impact of the expected faults on the services provided by the system. The final result of the impact analysis is used as input for traditional evaluation methods such as FMEA (Failure, Modes, and Effects Analysis) or FTA (Fault Tree Analysis). One of the main advantages of the EPA is its ability to evaluate composite models exhaustively, i.e., to prove the correctness of the design within the validity limits of the model. However, as with all such methods, the uncertainty surrounding some elements of the methodology is a crucial problem.

There are several reasons for this kind of uncertainty [3] originating in the insufficiency of the information incorporated into the composite model:

- The phenomenon of *error propagation* itself may be non-deterministic; for instance, if the propagation of data errors is influenced by intermediate data operations and faithful modeling of these is impossible at the given level of abstraction or would require an overly detailed model.
- Occasionally, especially in security analysis, faults and attacks can change the *system's structure* in unforeseen ways. Typical examples include parasitic couplings between functionally independent elements in the system or side-channel attacks.
- A similar problem is an impact assessment of emerging and *future security attacks*, which are unknown at the time of analysis.
- Finally, a significant source of uncertainty is the environment, as it is out of the system's control.

As the examples above show, the fundamental contradiction of impact analysis is that a precise and exhaustive method running over uncertain models can no longer guarantee a clear proof of correctness unless there is a proper solution to handle the uncertainty.

There are numerous solutions for dealing with uncertainty, such as probabilistic, fuzzy, and Bayesian network-based approaches quantifying the uncertainty and the final analysis results.

This paper presents a rough set theory-based (RST) uncertainty handling approach that delivers a pure qualitative view and avoids hard-to-estimate quantification. Our approach supports uncertainty handling in a qualitative, symbolic way, which allows the integration of the approach with existing discrete validation and verification (V&V) methods. Furthermore, using a qualitative approach, logic inference can be used to evaluate the system in an iterative way to reduce uncertainty gradually.

The RST-based error propagation analysis approach can be used to deal with both *epistemic* (knowledge-related) and *aleatory*-type (dependence on a random event) uncertainties. It can identify the critical points in the system whose fault can lead to a subsequent critical failure.

The paper first introduces RST (Sec. II) and EPA (Sec. III) as the basis of the uncertainty-aware EPA approach (Sec. IV). The paper continues with a use cases section (Sec. V) that explores the handling of both epistemic and aleatory uncertainties and concludes with a summary and future work description (Sec. VI).

II. Rough Set Theory

Pawlak [4] proposed RST, a mathematical paradigm to deal with imprecise, inconsistent, incomplete, uncertain information and knowledge. This section presents the basic concepts [5], [6] of RST.

An *information system* is a pair $IS = (U, A)$, where U is a non-empty finite set of objects called the *universe* and A is a non-empty finite set of *attributes* such that: $a : U \to V_a$ for every $a \in A$ The set V_a is called *value set* of a.

Decision systems are particular information systems with a distinguished attribute called the *decision attribute*. The decision attribute expresses a previously known decision (classification) based on the attributes. Formally, $DS = (U, A \cup d)$, where $d \notin A$ is the decision attribute.

Indiscernibility: Given a subset of the attribute set $B \subseteq A$ an indiscernible relation $IND(B)$ on the universe U can be defined as the element pairs which provide identical values for the potentially restricted set of attributes:

$$IND(B) = \{(x, y) | (x, y) \in U^2, \forall_{b \in B}(b(x) = b(y))\} \quad (1)$$

The *equivalence class* of an object $x \in U$ contains all the objects from them x is indiscernible. It is denoted by $[x]_{IND(B)}$ or shortly $[x]_B$.

Approximation sets: Given an information system $IS = (U, A)$, for a subset $X \subseteq U$ and $B \subseteq A$, two approximation sets are defined in the following way: *Upper (or brave)* approximation covers all the potential members of the set:

$$\overline{B}(X) = \{x \in U | [x]_B \cap X \neq \emptyset\} \quad (2)$$

meaning that $[x]_B$ has members in X.

Lower (or cautious) approximation covers all certain members of the set:

$$\underline{B}(X) = \{x \in U | [x]_B \subseteq X\} \quad (3)$$

meaning that all of $[x]_B$ members are in X.

This way, $\underline{B}(X) \subseteq X \subseteq \overline{B}(X)$.

The approximations partition the universe of the objects into three disjoint regions according to the decidability of containment in X:

$$POS(X) = \underline{B}(X), \text{the } \textit{positive} \text{ region} \quad (4)$$

$$BND(X) = \overline{B}(X) - \underline{B}(X), \text{the } \textit{boundary} \text{ region} \quad (5)$$

$$NEG(X) = U - \overline{B}(X), \text{the negative region} \quad (6)$$

- If an object $x \in POS(X)$, then it certainly belongs to target set X
- If an object $x \in BND(X)$, then it is undecidable whether the object x belongs to target set X or not.
- If an object $x \in NEG(X)$, then it certainly doesn't belong to target set X.

One main advantage of RST is the support of symbolic computations if the different approximations are defined by their respective membership functions. There are several rules supporting logic reasoning on rough sets [5], such as union, intersection, difference, and complementary with the usual manipulation of the respective membership functions.

Union:

$$\overline{B}(X \cup Y) = \overline{B}(X) \cup \overline{B}(Y) \quad (7)$$

$$\underline{B}(X \cup Y) \supseteq \underline{B}(X) \cup \underline{B}(Y) \quad (8)$$

Intersection:

$$\underline{B}(X \cap Y) = \underline{B}(X) \cap \underline{B}(Y) \quad (9)$$

$$\overline{B}(X \cap Y) \subseteq \overline{B}(X) \cap \overline{B}(Y) \quad (10)$$

Implication:

$$X \subseteq Y \implies \underline{B}(X) \subseteq \underline{B}(Y) \text{ and } \overline{B}(X) \subseteq \overline{B}(Y) \quad (11)$$

III. Error Propagation Analysis

The core idea of EPA is the simultaneous tracing of the value propagation of the same input over a fault-free system model and several faulty mutations according to the anticipated fault set. The diagnostic model for the EPA is shown in Fig.1.

- The reference (fault-free) component describes the intended behavior of the component. It takes the reference input and processes it according to the normal operational mode.
- The mutated component is based on mutation generation that happens by selecting the particular fault mode of a component in a generalized mutation metamodel.

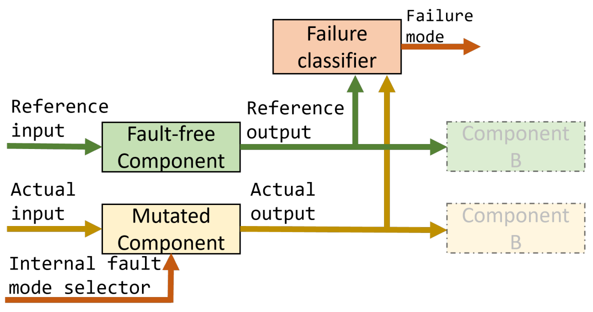


Fig. 1: Diagnostic model

The calculation of the actual output consists of two steps.

1) First, it depends on the actual input of the component, which can be either a previously processed value from a preceding component or an injected value (fault injection).
2) Second, the component processes the actual input regarding the selected fault mode. A fault selector selects the component's fault mode. In the case of the "No internal fault" assumption, the component processes the input as it internally works as intended. If an internal fault is selected, component mutations describe faulty operational modes.

Finally, failures can be estimated by comparing the reference output of the fault-free model with the particular outputs of the mutations. The failure classifier component determines the fault mode by comparing the reference and the actual output of a component. The failure classifier uses the semantics of the faults as predefined rules to determine the failure mode. For example, if the input and output have different values, it is a value difference data error. Naturally, the functionality of the failure classifier can be extended in a problem-dependent way. For instance, to estimate the severity of the failure.

The output of EPA is a table (e.g., Tab. I) containing scenarios in the rows. A scenario can be described by the active fault modes (fault activation) of the components and the violated requirements.

IV. Uncertainty-aware EPA

The result of the EPA is not always consistent and certainly complete due to the uncertainties mentioned in Sec. I. The resulting evaluation table can be interpreted as a decision system $DS = (S, FM \cup R)$, where S is a non-empty finite set of scenarios (as objects), FM is a non-empty finite set of possible component fault modes (as attributes), and R is a non-empty finite set of requirements (as decision variables). Table I is a decision system that contains 8 objects (S1-S8) with 2 attributes (FM1, FM3) and two decision attributes (R1, R2). Requirements can be formed based on the system's safety, security, or functional requirements.

For violated requirements (dangerous case), the corresponding objects (scenarios) can be selected $X = \{r | R_i(r) = Violated\}$. The RST approximation of the selected scenarios has the following interpretations:

- The upper approximation ($\overline{FM}(X)$) contains the scenarios with the fault activations which may lead to a violation (set).
- The lower approximation ($\underline{FM}(X)$) contains the scenarios with the fault activations, which certainly lead to a violation (set).
- $\overline{FM}(X) = \emptyset$. No requirement violation exists (if all the scenarios are sound and complete, it can be stated that the evaluated requirements cannot be violated in the original system). This proves the fulfillment of all evaluated requirements by the system.
- $\underline{FM}(X) \neq \emptyset \rightarrow \exists solution$, the existing solution will provide an example fault mode set and behavioral trace that leads to the requirement violation proving the insufficiency of the technical system.
- $BND(X)$, fault mode combinations in the boundary region are necessary but not sufficient conditions for the violation of the evaluated requirement, indicating suspicious systems suspected for further evaluation.

Rules interpreted on RST approximations (Rule 7-11) facilitate the fusion and joint investigation of information from different sources. For instance, if we have two independent requirements in the EPA, one is described by a technical sheet of a hardware component, and the other is a software-related requirement. We want to determine for which active (component) fault modes the occurrence of simultaneous violations of both requirements are either certain (lower approximation) or possible (upper approximation). The RST rule 9 obtains the joint lower approximation solution as $\underline{B}(X \cap Y) = \underline{B}(X) \cap \underline{B}(Y)$. Rule 10 obtains the estimate of the joint upper approximation: $\overline{B}(X \cap Y) \supseteq \overline{B}(X) \cap \overline{B}(Y)$.

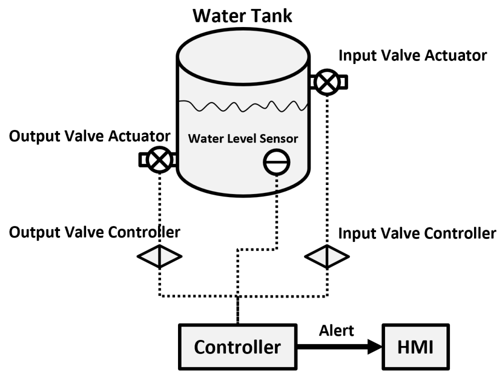


Fig. 2: Water tank architecture

When two system requirements are not mutually independent ($RX \subseteq RY$, where RX is a more specific requirement), then RST Rule 11 derives a containment relation for their respective lower and upper approximations ($\underline{B}(RX) \subseteq \underline{B}(RY)$ and $\overline{B}(RX) \subseteq \overline{B}(RY)$).

V. Use cases

The paper presents the use cases on a pilot study from [7]. The *water tank system* (Fig. 2) consists of a main water tank component with *input and output valve actuators* and their respective *controllers*. The water tank includes a *water level sensor* that measures the water level in the tank. The water tank *controller* sends control messages to the valve controllers based on the measure of the water level sensor. The system also contains a *Human-Machine Interface* (HMI) where the operator can see the status of the system and can react to dangerous events.

The *safety requirements* for the system are 1) `R1`: the water tank should not be overflow; 2) `R2`: alert should be sent to the operator in case of water tank overflow. The subsequent examples consider the following possible fault modes of the components: 1) `FM1`: Input Valve Stuck-at-Open 2) `FM2`: Output Valve Stuck-at-Open 3) `FM3`: HMI No signal

A. Epistemic uncertainty

The first example illustrates the handling of epistemic uncertainty. In Table I, there are many contradictions between the scenarios when examining the Requirement R1. Contradiction means here that for a given fault mode combination, the violation of the requirements is not unequivocal (e.g., for S4 and S5).

The first requirement is violated in Scenario S2, although none of the failure modes is active. The modeling incompleteness probably originates from some hidden, unknown state variable influencing R1.

Fig. 3 shows the negative region (NEG) in red, boundary region (BND) in blue, and positive (POS) region in green. The lower approximation for Requirements R1 and R2 are $\underline{FM}(R1) = \{S2\}$ and $\underline{FM}(R2) = \{S6, S8\}$. Their joint

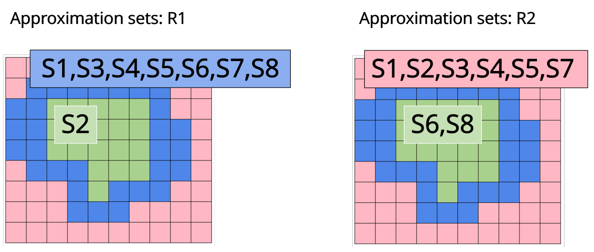


Fig. 3: Approximation sets for Table I

TABLE I: EPA evaluation table

Scenario	**Fault modes**		**Requirements**	
	FM1	**FM3**	**R1**	**R2**
S1	Active		-	-
S2			Violated	-
S3		Active	-	-
S4	Active		Violated	-
S5	Active		-	-
S6		Active	Violated	Violated
S7	Active	Active	-	-
S8	Active	Active	Violated	Violated

lower approximation yields an empty set ($\underline{FM}(R1 \cap R2) = \emptyset$). The joint upper approximation is $\overline{FM}(R1 \cap R2) = \{S6, S8\}$. As the lower approximation is empty, the boundary region equals the upper approximation, and no certain diagnostic conclusions can be drawn.

The purely uncertain result may originate from insufficient knowledge of diagnostics. The analyst can adaptively complement the analysis with sequential diagnostics by adding new observations to reduce epistemic uncertainty.

Adding a new observed attribute (fault mode) $FM2$ to the model (Tab. II) reduces but does not eliminate the uncertainty, as the boundary region is still non-empty. The next subsection further examines the remaining uncertainty in the analysis.

B. Aleatory uncertainty

Aleatory uncertainties can arise from model imperfections and non-deterministic behavior. This type of uncertainty is typical when a system contains third-party black-box components, has unknown parameterization (e.g., shared cloud system), or is a multiprocessor system. Investigating Requirement R1 (Table II) indicates a contradiction for the same anticipated fault set (stuck-open input and output valves $(FM1, FM2)$) between scenarios S4 and S5. Scenario S4 leads to a violation of Requirement R1, while Scenario S5 does not.

TABLE II: Extended evaluation table

Scenario	**Fault modes**			**Requirements**	
	FM1	**FM2**	**FM3**	**R1**	**R2**
S1	Active			-	-
S2		Active		Violated	-
S3			Active	-	-
S4	Active	Active		Violated	-
S5	Active	Active		-	-
S6		Active	Active	Violated	Violated
S7	Active		Active	-	-
S8	Active	Active	Active	Violated	Violated

This may be because the system model is built from a general component catalog, not binding by default all parameters of the elements. Thus, EPA checks all the individually existing but mutually exclusive combinations as if they were simultaneously present. In this specific case, the source of the contradiction could be the undefined relationship between the throughputs of the output valves:

- An output valve smaller than the input one may cause an overflow if both valves are stuck open (Scenario S4).
- If the relationship is opposite, the in-flowing water easily flows out of the tank through the larger output valve, and the requirement is not violated (in Scenario S5).

The joint lower bound for the simultaneous violation of Requirements R1, and R2 is: ($\underline{FM}(R1 \cap R2) = \{S2, S6, S8\} \cap \{S6, S8\} = \{S6, S8\}$). The lower bound can be transformed to a symbolic rule $(FM2(Active) \& FM3(Active) \implies Violation(R1, R2))$ describing the sufficient conditions of the evaluated requirement violations. This inductive reasoning-styled symbolic representation of sufficient (and necessary) conditions facilitates the explainable interpretation of the results for further validation of the system.

VI. Summary and Future Work

Rough Set Theory provides a powerful paradigm to assess and manage the effects of uncertainty at a global level. In many cases, however, the starting point for uncertainty is some local effect or lack of knowledge. In such cases, sensitivity analysis allows refinement of the model that results in global uncertainty by applying requirements limiting local uncertainty.

It should be emphasized that the primary mechanism of sensitivity analysis is essentially the same as that of EPA. It can be mapped into local information or system model faults and assessed their impacts.

References

[1] A. Pataricza, "Model-Based Dependability Analysis. DSc thesis." 2008.
[2] I. Kocsis, "Qualitative models in resilience assurance. Ph.D. dissertation," 2019, http://hdl.handle.net/10890/13122.
[3] M. Yazdi, S. Kabir, and M. Walker, "Uncertainty handling in fault tree based risk assessment: State of the art and future perspectives," *Process Safety and Environmental Protection*, vol. 131, pp. 89–104, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957582019305555
[4] Z. Pawlak, "Rough sets," *International journal of computer & information sciences*, vol. 11, no. 5, pp. 341–356, 1982.
[5] S. Akama, T. Murai, and Y. Kudo, "Reasoning with rough sets," *Logical Approaches to Granularity-Based Framework*, 2018.
[6] Q. Zhang, Q. Xie, and G. Wang, "A survey on rough set theory and its applications," *CAAI Transactions on Intelligence Technology*, vol. 1, no. 4, pp. 323–333, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2468232216300786
[7] A. Földvári, G. Biczók, I. Kocsis, L. Gönczy, and A. Pataricza, "Impact assessment of it security breaches in cyber-physical systems," in *2021 10th Latin-American Symposium on Dependable Computing (LADC)*. IEEE, 2021, pp. 1–4.

N-Version Programming as a Mitigation for Smart Contract Faults in Execute-Order-Validate Blockchain Systems

Bertalan Zoltán Péter, Imre Kocsis
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
bpeter@edu.bme.hu, ikocsis@mit.bme.hu

***Abstract*—In this paper, we propose the application of a well-known runtime fault-tolerance technique, N-Version Programming (NVP), as a new tool of smart contract software fault mitigation, especially for execute-order-validate blockchain systems, such as Hyperledger Fabric (HLF). Two patterns for aligning the NVP concept with the HLF architecture are proposed. A fully transparent solution where all peers have the same N versions installed and one we termed 'O-Version Programming' (where 'O' stands for 'Organization'), which relies on the majority voting aspects of execute-order-validate consensus mechanisms.**

***Index Terms*—blockchain, distributed ledger technology, n-version programming, runtime fault-tolerance, smart contracts**

I. Introduction

The software faults of smart contracts installed on blockchain systems may have devastating consequences. This has been made clear by unfortunate events such as the infamous Decentralized Autonomous Organization (DAO) hack in 2016 when 60 million US dollars worth of Ether was stolen [1]. In cryptocurrency contexts, faults result in potential financial loss. In consortial networks, however, the stakes may be much higher. For example, in a critical system application, smart contract faults might result in an accident or threat to human life.

Much research has been done to combat this problem, but it mainly focused on development-time fault-removal techniques, such as mutation testing [2], static analysis [3], and vulnerability detection via Machine Learning (ML) [4], to name a few. However, fault-tolerant computing also traditionally knows and employs a set of *runtime* patterns. These are problematic to apply to public blockchains such as Ethereum because the increased smart contract time and space complexity incur additional, significant costs[1]. There is no such problem with private, consortial blockchain platforms, where it may be reasonable to implement runtime fault-tolerance methods, even at the cost of higher complexity.

This paper concentrates on a single runtime pattern: N-Version Programming (NVP) [5]. NVP calls for the independent development of $n \geq 2$ functionally equivalent implementations of the same software specification. It is crucial that these implementations be as diverse as possible. To this end, they are often developed by different teams in different programming languages and using different algorithms and design patterns. The premise is that these different versions will likely not contain the same potential faults [6]. In the most elementary setup, the versions are executed in parallel, and their results are compared by a voter component which performs, for example, majority voting on them. In more complex applications, there may be an entire N-Version Execution Environment that manages the versions and their execution. NVP's popularity has somewhat dropped in recent years because some research suggests that the assumption of fault independence among the versions may be misguided or at least that care must be taken to consider dependent faults [7]. On the other hand, it seems that NVP might have a renaissance in the world of Artificial Intelligence (AI), as new publications suggest using it to improve reliability and resilience in ML models. The idea is to overcome the difficulty of reliable ML models by generating n versions of an ML component and then executing these diverse replicas, which costs more computations, but optimally results in significantly higher reliability [8], [9], [10], [11].

We identify two potential ways of applying NVP to smart contracts, especially in Hyperledger Fabric (HLF) (version 2.4). The first one is transparent from the platform's perspective and involves installing n versions of the same chaincode (or, more generically, smart contract) on every affected peer. The second approach takes advantage of the voting-like characteristics of HLF's consensus protocols. In the latter case, we propose that each organization have its own version installed on its peers, hence the coined name 'O-Version Programming (OVP)' ('O', as in 'Organization').

The rest of this paper is organized as follows: the next section compares our work with state-of-the-art fault-tolerance techniques for smart contracts. Then, section III and section IV introduce our classic, transparent approach and 'O-Version Programming', respectively. Finally, we summarize our work and describe its planned future advancements in section V.

[1]Smart contract deployment costs depend on the size of the contract. Then, each execution costs the calling party *gas,* which depends on the transaction complexity.

II. Related Work

To the best of our knowledge, there is currently only one application proposal of NVP in the context of Distributed Ledger Technology (DLT) and blockchain systems: *Hydra* [12], which uses a variant of NVP called N-of-N-Version Programming (NNVP) and focuses on error *detection* and safe termination rather than fault *tolerance* (the goal of classic NVP) featuring a bounty system rewarding finders of critical software faults. Technically, *Hydra* is designed for *Ethereum*. In contrast, our proposals *do* target the classic NVP objective of fault tolerance and focus on consortial DLT platforms, such as HLF.

III. Classic Approach

The most straightforward method to adopt NVP for *Fabric* chaincode (on first sight) is to simply install not one, but n implementations of the same chaincode specification everywhere (i.e. on every peer where the single chaincode would normally be installed) and then ensure that each version is executed, the results are compared, and some business logic decides the end result. This is the bare minimum, but more valuable features can be added, such as an additional layer of input and output validation before and after the versions are executed.

An arguably more elegant and viable alteration of this method is to package the entire NVP architecture into a single chaincode container, so in the perspective of *Fabric*, only a single chaincode is installed, and it can be invoked as usual. Behind the scenes, this chaincode is a facade hiding several chaincode versions and the validation and voting logic.

It is worth mentioning that *Fabric* also offers the capability to run chaincode as an external service rather than on the peers directly, opening up new possibilities for multi-version chaincodes. While external chaincode is not in the scope of this paper, it is a subject of future research regarding the topic.

A. *Master Chaincode as Controller*

If one wishes to follow the first elementary solution, the following steps must be taken:

1) Create a *master* chaincode as an entry point: it may perform input parameter checking, then invoke all n versions (passing on the input parameters), collect the returned values, compare them, and decide which (if any) result should be considered correct, and finally return that to the peer. This chaincode must also provide an Application Programming Interface (API) for reading and writing any data the chaincode needs, which the n versions will use.
2) Create n independent, diverse implementations of the chaincode specification. They may read and write the ledger via the master chaincode only.
3) Install all n versions as well as the master chaincode on all peers desired to be able to execute the chaincode.
4) Ensure clients are not allowed to interact with the n versions directly but only the master controller chaincode.

Figure 1 offers an architectural overview of this approach. The correct implementation of the master chaincode is essential, as it is a Single Point of Failure (SPOF) in the system. Fortunately, input/output validation and voting logic are not expected to be overly complex. *Fabric*'s permissioning system can control who may invoke what chaincode, so it is possible to forbid the invocation of the individual versions by any client.

One downside to this method is that $n{+}1$ chaincode versions must be installed and maintained separately, and *Fabric* has no way of knowing they are related. Furthermore, since every chaincode essentially has its own namespace, the chaincode variants are to perform read/write operations via the master chaincode to be able to access the same data. This complicates matters to an extent that makes the containerized approach outlined in the next subsection superior in almost every aspect. The benefit is that this kind of complete separation of the implementations makes it possible to mix programming languages easily, further increasing software diversity – something which is more complicated in the containerized architecture.

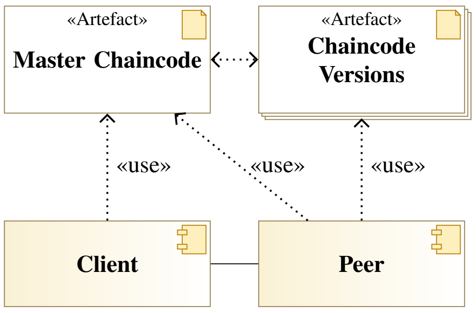


Fig. 1. Component diagram of the Master Chaincode-based approach

B. *Containerized Approach*

The fact that HLF chaincode runs in *Docker* containers makes it possible to design more complex architectures, such as parallel execution of code through multithreading. The idea is to package the entire $n+1$ versions into one self-contained unit that can be installed on a peer just like any regular chaincode.

Instead of exposing the peer's interface to the chaincode implementations so they can read the ledger contents, we propose providing a *proxy* that can cache ledger reads (since a single read of a ledger value is always sufficient, you cannot 'read-your-write'). The final read/write set is built by the NVP Controller (NVC) component based on the reads intercepted by the proxy and the writes suggested by the chaincode variants. In the simplest case, differing read-write sets may be considered erroneous. Alternatively, the NVC may implement logic that can combine the individual versions' read-write sets, provided they are compatible. A possible architecture can be seen in Figure 2. NVC and NVX refer to NVP Controller and NVP Executor, respectively.

One possible way to adopt this approach in software is by using *Java* threads and the *active object* pattern. Implementations of the same chaincode specification interface are known by the controller class. After ensuring the validity of the input, the implementations are executed in parallel.

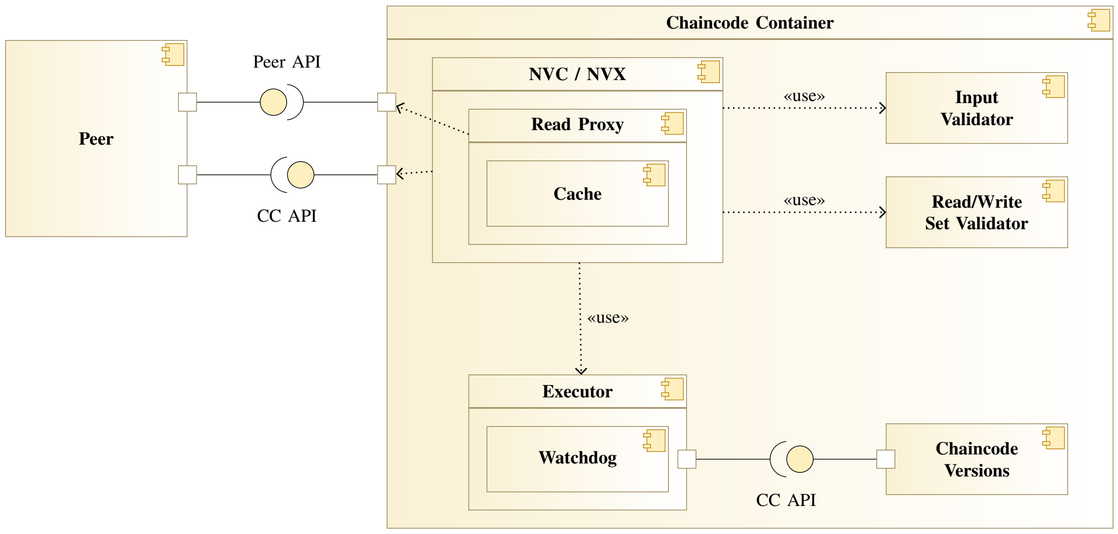


Fig. 2. Component diagram of the containerized approach

IV. O-Version Programming

Instead of installing multiple chaincode versions on the peers either directly or using the architecture outlined previously, it is possible to rely on the consensus mechanism of HLF. In this case, organizations or peers may have their own chaincode version installed independently. In some contexts, this might even be a requirement; for example, consider a weather forecast service where several clients attempt to submit their sensor data to the ledger. The supporting chaincode can be the organization's own as long as it implements the same specification as all others.

Contrary to the other approaches, there is no specialized voter component in this method. Deciding which versions' results are correct is deferred to the platform's consensus protocol: in the end, the configured endorsement policy determines the outcome. For example, for maximum fault tolerance (at the cost of low availability), given n peers hosting their own versions, an $n : n$ endorsement policy ensures that either every single implementation has the same active software fault (quite unlikely) or the correct results are appended to the ledger. Otherwise, the transaction is rejected. Note that we do not take special client application side logic into consideration, i.e. clients are assumed not to discard any endorsements.

Figure 3 contains an overview of this strategy: even if one of the chaincode variants is faulty, the policy of at least two endorsements needed ensures that at least one correct chaincode is executed for each transaction, producing disagreeing results.

A. *Effects on Attack and Fault Tolerance*

An interesting aspect of this approach is that it enhances the ledger integrity-preserving role of endorsement; usually, all peers have the same single implementation of the chaincode, and endorsement ensures that no malicious organization or peer can alter ledger contents to their advantage. In the following, we examine how this *standard* function of endorsement is affected by OVP.

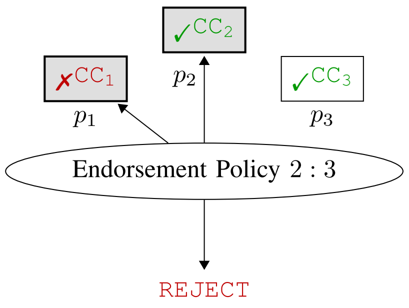


Fig. 3. Overview of consensus-based NVP

Consider a network of six organizations: three are running version A of a chaincode specification, and the other three are running version B. If we assume one of the two versions faulty, it follows that at least four organizations' endorsements should be required to ensure integrity. This way, even if all three organizations with the faulty chaincode version propose to append the same faulty transaction to the ledger, the additionally required fourth organization, which certainly has a fault-free implementation, will prevent the undesired ledger update. Figure 4 offers a visualization of this example for better understanding.

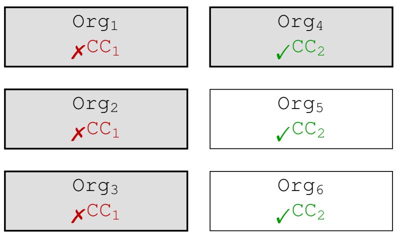


Fig. 4. Two chaincode versions distributed among six organizations

Optimal Endorsement Policy Based on Diversity, Faults, and Malicious Organizations: For simplicity, let us assume, for the remainder of this section, that every organization only maintains a single peer node (in reality, the number of peers per organization is an additional factor to consider). The question the answer to which we seek is the following: given $n \in \mathbb{N}^+$ peers (any positive integer) and $v \in \{1\,..\,n\}$ chaincode versions, what $k : n$ endorsement policy is required (expressed as bestk) to tolerate $m \in \{0\,..\,n-1\}$ malicious peers if $f \in \{0\,..\,v-1\}$ of the versions are faulty? We make the following observations:

- In any case, at least one endorser is necessary, so bestk is at least 1.
- Any m number of malicious parties implies the requirement of m more endorsements.
- For an f number of faulty chaincode versions, the policy must be altered to include f times the number of peers who have that version, which is $^n/_v$ (assuming a uniform distribution).

Based on the above, Equation 1 can be obtained as a function of four variables. For the sake of visualization, we combine the values of n and v into a single variable $r = {}^v/_n$ we call the *diversity ratio* – Figure 5 shows a 3D surface plot of the thus obtained function. The plot has been made with a fixed zero value for the number of malicious organizations, but different values of m would merely shift the plot along the z (vertical) axis. As for the other input values, $n, v \in \{1\,..\,10\}$ (therefore $r \in \{^1/_{10}, {}^2/_{10}, \ldots, 1\}$) and $f \in \{0\,..\,9\}$.

$$\mathrm{bestk}(n, v, f, m) = 1 + m + f\frac{n}{v} \tag{1}$$

The result shows that for low diversity ratios and a high amount of software faults, very intolerant endorsement policies are necessary, as expected. With higher chaincode diversity, the plot flattens. Software faults have a much higher impact on the necessary endorsement policy than the number of malicious organizations to tolerate – this makes sense, as several organizations might be running the same version.

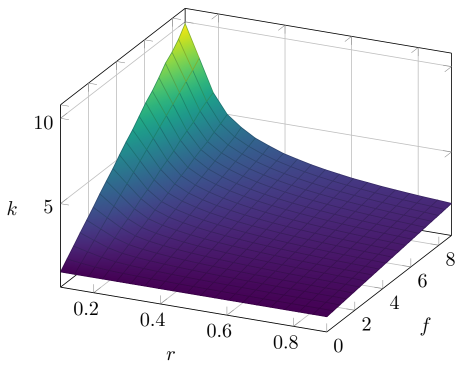


Fig. 5. 3D surface plot of the bestk function

V. Conclusion and Further Work

In our work, we proposed revisiting classic N-Version Programming to address the impact of software faults in chaincode, which can increase fault tolerance by means of increasing software diversity. We have presented two entirely different ways of its integration: a ‘classic’ approach and ‘O-Version Programming.’ As the latter builds on the same mechanism of the network, which ensures its high integrity, consensus, we briefly analyzed its impact by observing how the number of chaincode versions and the number of software faults interplay with the endorsement policy and the number of malicious participants in the network.

We conclude that when consensus is used for NVP, the choice of endorsement policy mostly depends on the number of chaincode versions and the number of peers. We have offered an architectural design for the ‘classic’ style (instead of relying on consensus, each peer has the same n versions), which we would like to prototype as future work to demonstrate its viability. There may be additional models of NVP implementations, which are yet to be explored; for example, using *Fabric* v2’s external chaincode service features.

References

[1] V. Dhillon, D. Metcalf, and M. Hooper, *The DAO Hacked*. Berkeley, CA: Apress, 2017, p. 67–78.

[2] J. J. Honig, M. H. Everts, and M. Huisman, “Practical mutation testing for smart contracts,” in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, C. Pérez-Solà, G. Navarro-Arribas, A. Biryukov, and J. Garcia-Alfaro, Eds. Cham: Springer International Publishing, 2019, p. 289–303.

[3] J. Chen, X. Xia, D. Lo, J. Grundy, X. Luo, and T. Chen, “Defectchecker: Automated smart contract defect detection by analyzing EVM bytecode,” *IEEE Transactions on Software Engineering*, vol. 48, no. 7, p. 2189–2207, 2022.

[4] J. Zhang, L. Tu, J. Cai, X. Sun, B. Li, W. Chen, and Y. Wang, “Vulnerability detection for smart contract via backward bayesian active learning,” in *Applied Cryptography and Network Security Workshops*, J. Zhou, S. Adepu, C. Alcaraz, L. Batina, E. Casalicchio, S. Chattopadhyay, C. Jin, J. Lin, E. Losiouk, S. Majumdar, W. Meng, S. Picek, J. Shao, C. Su, C. Wang, Y. Zhauniarovich, and S. Zonouz, Eds. Cham: Springer International Publishing, 2022, p. 66–83.

[5] A. Avizienis, “The n-version approach to fault-tolerant software,” *IEEE Transactions on Software Engineering*, vol. SE-11, p. 1491–1501, 01 1986.

[6] ——, “The methodology of n-version programming,” *Software fault tolerance*, vol. 3, p. 23–46, 1995.

[7] J. C. Knight and N. G. Leveson, “An experimental evaluation of the assumption of independence in multiversion programming,” *IEEE Transactions on Software Engineering*, vol. SE-12, no. 1, p. 96–109, 1987.

[8] A. Gujarati, S. Gopalakrishnan, and K. Pattabiraman, “New wine in an old bottle: N-version programming for machine learning components,” in *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2020, p. 283–286.

[9] H. Xu, Z. Chen, W. Wu, Z. Jin, S.-y. Kuo, and M. Lyu, “NV-DNN: Towards fault-tolerant DNN systems with n-version programming,” in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2019, p. 44–47.

[10] F. Machida, “N-version machine learning models for safety critical systems,” in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2019, p. 48–51.

[11] A. Wu, A. H. M. Rubaiyat, C. Anton, and H. Alemzadeh, “Model fusion: Weighted n-version programming for resilient autonomous vehicle steering control,” in *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2018, p. 144–145.

[12] L. Breidenbach, P. Daian, F. Tramèr, and A. Juels, “Enter the hydra: Towards principled bug bounties and exploit-resistant smart contracts,” in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, 08 2018, p. 1335–1352. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/breindenbach

Knowledge-driven Exploratory Performance Data Analysis for Execute-Order-Validate Blockchains

Noor Al-Gburi
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
noor.algburi@edu.bme.hu

Imre Kocsis
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
kocsis.imre@vik.bme.hu

***Abstract*— Exploratory data analysis (EDA) of the performance characteristics of complex IT systems, such as enterprise blockchain solutions, would significantly benefit from explicit representations of, and inference on knowledge about the analyzed system. However, connecting EDA and knowledge representation is not part of the current practice. As a novel approach, this paper presents a generic hierarchical activity ontology, connected to Hyperledger Fabric experiments with end-to-end delay, endorsement delay, ordering delay, and block validation observations. On this basis, we present rules for inferring knowledge-based visualization declarations on this ontology. Lastly, we generate Jupyter notebooks for the inferred sequence of visualizations.**

Keywords—Ontology-based approach, Hyperledger Fabric, Knowledge-based, Exploratory Data Analysis.

I. Introduction

The performance characteristics of complex IT systems heavily rely on empirical methods. Exploratory Data Analysis (EDA) [1] is fundamental to performing data analysis. EDA is a highly visual process where the investigation process of data is not prescribed; rather, a few best practices complement the intuition of the data analyst. However, for truly complex systems, described through many observational variables, simple intuitive "detective work" becomes inefficient and intelligent guidance becomes necessary. Approaches such as grand tours [2] exist, but these consider only the statistical properties of the data, and not its inherent structure related to the described processes.

On the other hand, recently, significant progress has been made in the knowledge-based characterization of the observable performance properties of IT systems [3]. The core idea of this research is to utilize such explicit knowledge representations for the sake of guiding EDA. The underlying intuition is that data analysts tend to have a "default style" of EDA, either by personal preference or motivated by domain knowledge, which primarily depends on the *type of the described processes* and not the data.

Thus, for a given system class, for instance, hierarchically composed parallel-sequential task executions, there is a "sensible" progression of "usual" plot types, which can be *inferred* from the composition of the processes.

In this paper, we present an elaboration of this idea which, to the best of our knowledge, has not been tackled yet in the literature.

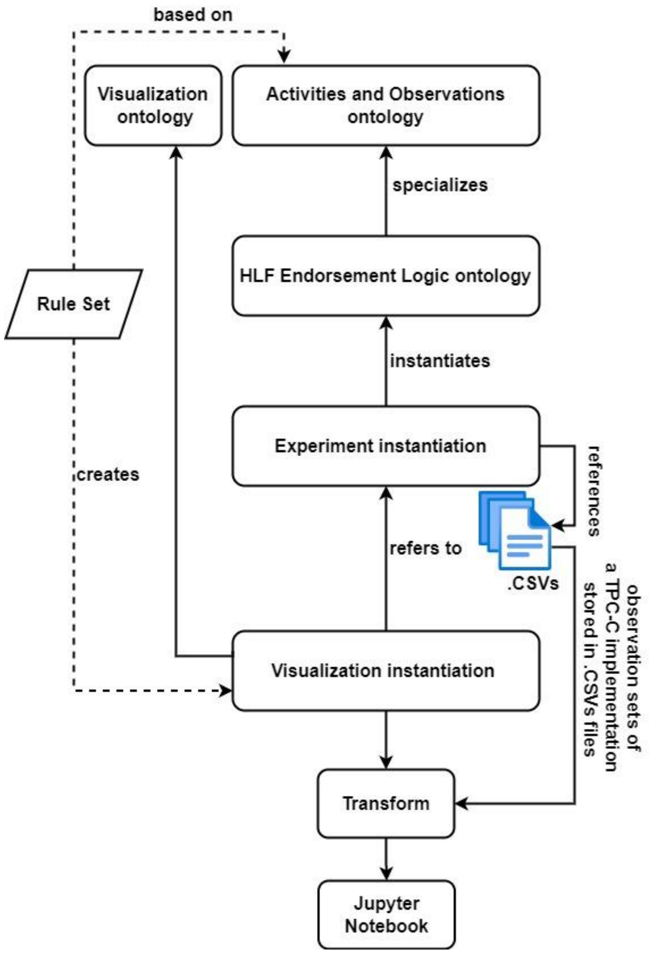
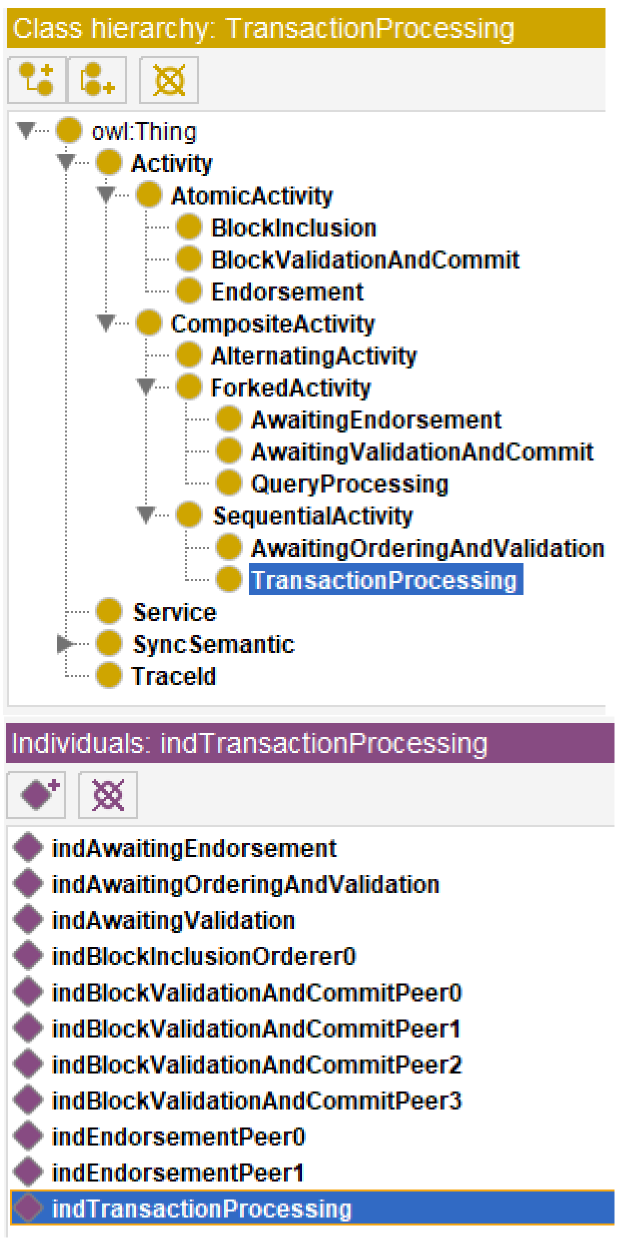

based on
Visualization ontology
Activities and Observations ontology
specializes
HLF Endorsement Logic ontology
Rule Set
instantiates
Experiment instantiation
references
creates
refers to
.CSVs
observation sets of a TPC-C implementation stored in .CSVs files
Visualization instantiation
Transform
Jupyter Notebook

Figure 1. Overview of the proposed approach

An existing ontology for hierarchically composed parallel-sequential task executions [4] serves as an abstract observed behavior class. We created a specialization of this ontology, which describes the hierarchically composed parallel-sequential tasks performed during a part of the consensus


Class hierarchy: TransactionProcessing

- owl:Thing
  - Activity
    - AtomicActivity
      - BlockInclusion
      - BlockValidationAndCommit
      - Endorsement
    - CompositeActivity
      - AlternatingActivity
      - ForkedActivity
        - AwaitingEndorsement
        - AwaitingValidationAndCommit
        - QueryProcessing
      - SequentialActivity
        - AwaitingOrderingAndValidation
        - TransactionProcessing
  - Service
  - SyncSemantic
  - TraceId

Individuals: indTransactionProcessing

- indAwaitingEndorsement
- indAwaitingOrderingAndValidation
- indAwaitingValidation
- indBlockInclusionOrderer0
- indBlockValidationAndCommitPeer0
- indBlockValidationAndCommitPeer1
- indBlockValidationAndCommitPeer2
- indBlockValidationAndCommitPeer3
- indEndorsementPeer0
- indEndorsementPeer1
- indTransactionProcessing

Figure 2 Concepts of the endorsement and transaction processing ontology with individuals

process in the blockchain platform Hyperledger Fabric (HLF).

We extend this ontology so that it can reference specific observation sets of a TPC-C implementation over Fabric [5], stored in .csv files.

As a next step, we propose a rule based on *the abstract ontology* to determine a "sensible sequence" of plots for performing EDA *on the specific Fabric observational data*. The plots are created by the rules as a set of "plot individuals" in the ontology domain, ordered by EDA plot sequence ordering properties.

Finally, a pluggable "last mile" step is proposed to translate the EDA plot individuals and their ordering to the specific technology used for EDA; currently, Python cells in notebooks. Fig. 1 depicts the proposed approach.

The remainder of the paper is structured as follows. Section 2 introduces an overview of using knowledge bases to describe visualization and the related work. Section 3 discusses the trace processing and the endorsement visual analytics of Hyperledger. Section 4 presents the rule-based inference of visualization. Section 5 concludes this paper.

II. Knowledge-based Characterization of Visualization

Visual exploration of empirical data is a proven tool in setting up extra-functional behavioral models for IT systems, as well as diagnosing them in comparison to explicit as well as implicit models of expected behavior [6]. However, making such analytic endeavors efficiently repeatable is an *open research problem*. Certainly, for a given type of system and given analytic goal, data analytic templates (e.g., analysis notebooks) can be created, but these neither provide insight into the implemented process readily nor are easily adaptable to insights found during the process or changes in the analyzed system. Another challenge is that tying computer-aided exploration suggestions – classically techniques such as projection pursuits and grand tours, nowadays "autoML" methods – into the process is not straightforward.

Intuitively, what can be suspected as the underlying problem is the lack of explicit formulations, and reasoning about, the visual analytic process of IT system observations itself. Supporting this hypothesis in a broader context, recently, there have been calls [7] for creating explicit theories of (graphical) inference, to transform EDA into an activity which looks for deviation from some "norm" (the challenge is, certainly, that the "norm" is in most cases an implicit, ambiguous, and highly qualitative model in the mind of the analyst).

Following in the footsteps of data science process models (such as CRISP-DM), explicit ontology-based descriptions of the 'what" and "how" of visually assisted machine learning processes are also emerging [8]. Knowledge Graphs have been proposed to capture users' visual analytics workflows [9]. Using ontology-expressed knowledge to provide user guidance during visual analytics also has some prior art; however, as [10] points out, the guidance generation process is currently poorly understood and expressible in general terms.

Our initial inroad to the specific research problem here relies on a hierarchical *domain ontology*, which captures system behaviour in abstract terms as well as the specifics of actual system deployments.

III. Knowledge-based Characterization of Measurements

General semantic data analysis is an open research topic. It is challenging to automatically identify hidden relations among a general set of variables to aid later data analysis phases. However, given a specific domain and a priori knowledge of relations among domain concepts can benefit greatly from such domain-specific knowledge. [4] has shown how a semantic model of system activities and services with multiple levels of abstraction can guide measurement data correlation, calculation, and validation steps.

A. Knowledge-based trace processing for HLF

[4] uses the Hyperledger Fabric blockchain platform as an example. Blockchain is an essential technology that can decentralize the way we store, share and information and data. One of the newer blockchain platforms that emerged is

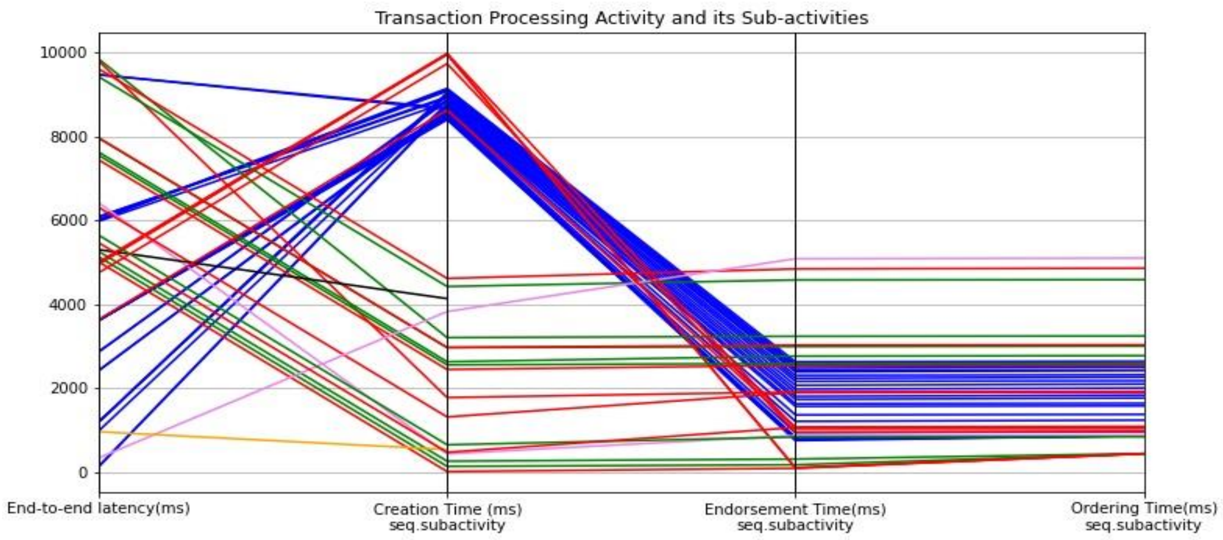


Figure 3. Visualization of the execution time of a transaction activity and its sequential refinement

Hyperledger Fabric, a project of the Hyperledger Foundation community. It is a global collaboration, hosted by The Linux Foundation.

HLF is an open-source blockchain platform [11]. Hyperledger Fabric is an implementation that enables permissioned blockchains, which provide a general blockchain framework with identifiable participants for a variety of business applications. In contrast to public networks, Hyperledger Fabric's performance is highly engineerable, which also means that network design decisions must be validated from the performance point of view.

Fabric supports the deployment of general-purpose smart contracts on the network nodes, called "*chaincode*", that can manipulate the shared state through atomic transactions. Network-level consensus begins with an "endorsement" phase (a client requests smart contract simulation from network peers). The collected endorsements are ordered and put into blocks by the so-called ordering service, which finishes consensus by distributing the blocks to network nodes for validation and incorporation into the blockchain. For further details, see [11].

In [4], a detailed activity model for distributed transaction processing (such as the HLF consensus process) facilitates the correlation and availability check of distributed activity traces. The activity model of the HLF case study defined the measured temporal data of activities, associated with the service types logging them. Accordingly, the prerequisite trace correlation step simply followed the structure of the model to check whether all supposedly measured data are available from all sources. The checks successfully revealed a number of data anomalies.

B. *Extensions for endorsement visual analytics*

The endorsement phase of HLF consensus is particularly important in HLF performance engineering [12]. We have created an ontology refinement to describe the endorsement phase of HLF consensus (including end-to-end transaction processing), represented as hierarchical sequential parallel activities from [4]. The main concepts are depicted on Fig. 2. In addition to describing a hierarchical sequential-parallel process, the ontology can host references to observations on the activities, stored in CSV files.

The model details the high-level, sequential steps of HLF end-to-end transaction processing. The first, *Awaiting Endorsement* activity has parallel sub-activities with the type endorsement (the entire transaction begins with the client getting an endorsement from the peers). *Awaiting Ordering and Validation* is modelled by two consecutive sub-activities: *Block inclusion* (that is an atomic activity) and the client *Awaiting Validation* from any peer (denoted by *any* synchronization semantic).

Instances of individuals represent the low level of the ontology that makes a relationship with identified data properties of the classes, for example; everything in the class hierarchy of the transaction processing has a unique identifier (ID); in this model we have individuals like the individual *indTransactionProcessing* that has a class of Transaction Processing, or the individual *indBlockInclusionOrdere0* which has a class of *Block Inclusion*. Moreover, we also have *indEndorsementPeer0* and *indEndorsementPeer1*: individuals which have a type of *Endorsement* (in this individual level will take one awaiting endorsement which has type *Awaiting Endorsement*), and four individuals for the validation indBlockValidationAndCommitPeer0/1/2/3.

IV. Rule-base inference of visualization

The simplest EDA-supporting framework for characterizing the performance (latency and throughput) of hierarchically composed parallel-sequential activities implements a breadth-first search over the refinement tree of the activities. In this refinement tree, questions regarding the relationship between the latencies and throughput of the refining and refined activities can be investigated. Fig. 3 provides an example of the hierarchical end-to-end latency breakdown visualization of transactions, as comprehended by the client.

Currently, we are investigating the use of SWRL [13] rules to infer a breadth-first search style progression of parallel coordinates plot *declarations* for any modelled HLF deployment (and annotation-connected observation set),

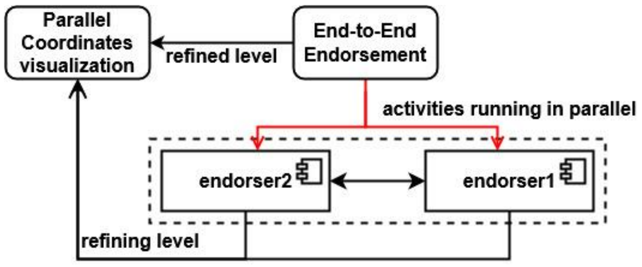


Figure 4. From activity refinement patterns to visualization

based on the activity refinement relations captured through the abstract ontology. Fig. 4 demonstrates the pattern-matching concept.

The role of the SWRL rules is to create new visualization objects, typed by a simple visualization ontology, which among themselves have a partial order relationship and indirectly point to the .CSV files and columns to be visualized. For example, we can add SWRL rules to suggest visualizations for every activity based on the structures of the ontology or add a data property to the abstract activity class. As a simple example, when investigating the hierarchical model, we can say that if there is an activity which is either a parallel, or forked activity and has sub-activities, then fill a data property with a "string flag" such as "*Create parallel Coordinates with children*":

```
hasSubactivity (?sub, ?sa)     ->
visualization_directive(?sub,  "Create
Parallel coordinates with children")
```

This enables us to distinguish what to plot after transposing these entities to a sequence of Python cells in a Jupyter notebook implementing the necessary data loading and the inferred visualizations.

V. Summary

Exploratory data analysis (EDA) of the performance characteristics of complex IT systems, such as enterprise blockchain solutions, would significantly benefit from explicit representations of, and inference on knowledge about the analyzed system. In this paper, we presented a specification of hierarchically composed parallel-sequential task executions and extended this ontology with an endorsement activity and transaction processing ontology to reference a specific set of observations of a TPC-C implementation over a Fabric network. SWRL rules will enable generating comparative visualization directives for each level of the composition, and their suggested analysis order. As a last step, the inferred series of visualizations – to support EDA – will be transposed to a Jupyter notebook.

The proposed approach, however, only addresses the challenge of rapidly creating built-in "styles" of EDA for wide ranges of HLF deployments. As a next step, our research will target coupling knowledge-based visualization inference with data-based intelligent *guidance* for EDA.

References

[1] J. T. Behrens, "Principles and Procedures of Exploratory Data Analysis," Psychol. Methods, vol. 2, no. 2, pp. 131–160, 1997.

[2] H. Wickham, D. Cook, H. Hofmann, and A. Buja, "tourr: An R Package for Exploring Multivariate Data with Projections," J. Stat. Softw., vol. 40, no. 2, pp. 1–18, Apr. 2011.

[3] A. Klenik and A. Pataricza, "Adding semantics to measurements: Ontology-guided, systematic performance analysis," Acta Cybern., pp. 1–36, 2021.

[4] A. Klenik, "Measurement-based performance evaluation of distributed ledger technologies," PhD. Dissertation, Budapest University of Technology and Economics, 2022.

[5] A. Klenik and I. Kocsis, "Porting a benchmark with a classic workload to blockchain: TPC-C on Hyperledger Fabric," Proc. ACM Symp. Appl. Comput., pp. 290–298, Apr. 2022.

[6] A. Földvári and A. Pataricza, "Semi-automated model extraction from observations for dependability analysis," in 2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2021, pp. 1–6.

[7] J. Hullman and A. Gelman, "Issue 3.3, Summer 2021," Harvard Data Sci. Rev., vol. 3, no. 3, Jul. 2021.

[8] D. Sacha, M. Kraus, D. A. Keim, and M. Chen, "VIS4ML: An Ontology for Visual Analytics Assisted Machine Learning," IEEE Trans. Vis. Comput. Graph., vol. 25, no. 1, pp. 385–395, Jan. 2019.

[9] B. Karer, I. Scheler, H. Hagen, and H. Leitte, "ConceptGraph: A Formal Model for Interpretation and Reasoning During Visual Analysis," Comput. Graph. Forum, vol. 39, no. 6, pp. 5–18, Sep. 2020.

[10] S. Miksch, H. Leitte, and M. Chen, "Knowledge-Assisted Visualization and Guidance," Found. Data Vis., pp. 61–85, 2020.

[11] E. Androulaki, A. Barger, V. Bortnikov, S. Muralidharan, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Murthy, C. Ferris, G. Laventman, Y. Manevich, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains," Proc. 13th EuroSys Conf. EuroSys 2018, vol. 2018–January, Apr. 2018.

[12] I. Kocsis, A. Klenik, A. Pataricza, M. Telek, F. Deé, and D. Cseh, "Systematic Performance Evaluation Using Component-in-the-Loop Approach."

[13] M. O'Connor, H. Knublauch, S. Tu, B. Grosof, M. Dean, W. Grosso, and M. Musen, "Supporting rule system interoperability on the semantic Web with SWRL," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 3729 LNCS, pp. 974–986, 2005.

Transfer Learning in Heterogeneous Drug-Target Interaction Predictions Using Federated Boosting

Dániel Sándor, Péter Antal
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
sandor@mit.bme.hu, antal@mit.bme.hu

***Abstract*—In multitask federated learning, when small amounts of data are available, it can be harder to achieve proper predictive performance, especially if the clients' tasks are different. However, task heterogeneity is common in modern Drug-Target interaction (DTI) prediction problems. As the data available for DTI tasks are sparse, it can be challenging for clients to synchronize the tasks used for training. In our method, we used boosting to enhance transfer in the multitask scenario and adapted it to a federated environment, allowing clients to train models without having to agree on the output dimensions. Boosting uses adaptive weighting of the data to train an ensemble of predictors. Weighting data boosting can induce the selection of important tasks when shaping a model's latent representation. This way boosting contributes to the weighting of tasks on a client level and enhances transfer, while traditional federated algorithms can be used on a global level. We evaluate our results extensively on the tyrosine kinase assays of the KIBA data set to get a clear picture of connections between boosting federated learning and transfer learning.**

***Index Terms*—federated learning, multitask learning, boosting, DTI**

I. Background

In the past years, drug discovery has become more and more reliant on the use of machine learning [1]. For a long time, drug discovery was mostly conducted with in vitro tests on candidates. But these types of tests are expensive and time-consuming to execute thus, they do not scale well for the large amounts of available drug candidates [2]. Machine learning has the advantage of scalability and relatively low resource costs thus, it can complement traditional tests well. Modern drug discovery problems usually involve a model which can select or narrow down a list of drug candidates which are further tested in laboratories [3].

The motivation for our work is a federated drug-target interaction problem. In this, we try to predict which chemical compounds (drugs) will bind to which biological targets (e.g. proteins) in the human body, to induce a change in the organism [4]. For this task, there is already a large amount of data available [5]–[7]. The historically used compounds of drugs usually have multiple targets to which they are likely to bind hence the only job is to find these targets. The data on targets is available through assays. Assays are procedures to analyze the qualities of a given target. In this case, this means the identification of drugs that bind or do not bind to any given target. A usual DTI task is built up in the following way: multiple assays are available for one target, and assays are only conducted for some compounds (and not all available) to save resources. After the collection of assays, they can be arranged into a bioactivity matrix, which contains one assay in a column and one compound in a row. Because the assays only have available data on interactions where they have been measured and the measurements are only conducted where the researchers saw a chance of interaction the resulting data is sparse and missing-not-at-random (MNAR) type data.

To explain the motivation of federated learning (FL), we have to look at the nature of the data. When working with high-value data, like data on drugs, companies can be reluctant to share it. Although in some cases the demand presents itself to learn from a larger data set and cooperate with other companies or research institutions. In these cases, privacy-preserving federated learning can be a useful tool for performing the training. In federated learning, multiple parties agree to train their models together for better performance, while their data sets remain private and not shared. This can be achieved in a multitude of ways. One of the most prominent examples is Federated Averaging (FedAvg) [8], which uses a shared model architecture and averages the models' weights every few iterations.

Multitask learning is a natural extension of this setup as it enhances the performance of models on targets when learned jointly [9]. However multitask learning also presents challenges, namely selecting tasks in a way to achieve the best possible performance [10]. To solve this we introduce an adaptive weighting by the use of boosting. This way at the beginning of training no task selection is necessary instead the data will be weighted in every iteration to maximize the performance of models.

Boosting is a highly versatile method, thus it can be used for drug research too. In [11] Svetnik et al. showed that tree-based boosting is especially useful in Quantitative structure-activity relationship (QSAR) problems, where the goal is to predict bioactivity from structural features, much like in DTI. However, in an FL scenario boosting is still a relatively

This research was funded by the J. Heim Student Scholarship, OTKA-K139330, the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory, New National Excellence Programme of the Ministry of Innovation and Technology, code number ÚNKP-22-2-I-BME-70, funded by the National Research, Development and Innovation Fund.

new concept and only a handful of solutions exist. Most of these are from the field of gradient boosting: One of these is SecureBoost [12], which is a framework for gradient boosting by decision trees, however, it only works in vertical FL scenarios, when the data is only distributed in the feature space, not in the sample space, which is a strong assumption. In [13] Li et al. create a similarity-based federation for gradient boosting, where each party makes their data public through hashing and they train models on the hashed features. These approaches are significantly different from AdaBoosting, and in the next chapter, we give a detailed explanation, of why AdaBoost is a good fit for the problem at hand. The adaptation of boosting to multitask can also be done in several ways: Wang et al. [14] created the Online Multitask Boosting (OMB) algorithm, which is the generalization of a transfer learning boosting algorithm. Their algorithm captures task relatedness by calculating the differences in errors of the ensemble on the two tasks. In their algorithm, some tasks are used to train models of the ensemble and every task learns to adapt them by assigning its own weight to the output of the classifier. A different approach is described by Zhang et al. [15] in a technique they call MTBoost. In MTBoost they learn the relationships of tasks in the form of a task covariance matrix. First, they learn an ensemble, for a base hypothesis and generate the output for every task by learning the weights specifically for them. The base hypothesis can be formulated by learning an aggregated "super-task" based on the task covariances.

In our work, we aim to combine the best of all these approaches in a way that makes it feasible to create a multitask ensemble of neural networks in a federated manner, which leverages the adaptive weighting of samples to boost predictive performance. To do this we create a combination of the FedAvg and the AdaBoost algorithms, using neural networks as base classifiers. We present our results on the KIBA data set, with a realistically constructed federated split for the data.

II. Method

To understand federated boosting, we start from the same setup as the FedAvg algorithm: Every client has their own data set, with possible overlaps in both the compound- and target spaces. In the beginning, the clients assign a uniform weight to each of their compounds: $\omega_{i0} = 1/N, i = 1, 2, \ldots, N$

After this, the server initializes the model weights w_t and distributes them to the clients. The clients run a local training sequence on the model, in my experiments, this means one epoch of training the network. Next, the clients send their models back to the server for aggregation (alternatively, like in FedAvg it is enough to send the gradients). The server averages the weights and produces the final model, which will be part of the ensemble.

$$w_t \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_t^k \tag{1}$$

After this, the model is sent back to the clients and they calculate the error of the model on their data using the formula from AdaBoost, with the difference being, that we classify a compound as correctly classified if a given threshold of correctness is reached on its predictions. In my measurements, this meant that 80% of a compound's measurements are to be predicted correctly for the compound to be classified as correct.

$$Error_t = \sum_{i=0}^{n} \omega_{i(t-1)} * I_{y_i \neq \hat{y}_i} \tag{2}$$

Based on the error, they assign an individual weight to the model (same as AdaBoost).

$$\alpha_t = \frac{1}{2} log(\frac{1 - Error_t}{Error_t}) \tag{3}$$

Now the partners know the weight of the model, they can reweight the data and start the training again with new weights from the server.

Algorithm 1 FedMTBoost

```
 1: **Initialization on the clients:**
 2: Initialize the observation weights $\omega_{i0} = 1/N$, $i = 1, 2, \ldots, N$
 3: initialization of ensemble $E = \{\}$
 4: initialization of $\eta$ learning rate
 5: **Computed on the server:**
 6: for $t = 1, 2, \ldots$ do
 7:     initialization of $w_t$
 8:     $S_t$ = set of clients
 9:     for $k \in S_t$ clients do
10:         Send $w_t$ to $k$
11:         Recieve $w_t^k$
12:     end for
13:     $w_t \leftarrow \sum_{k=1}^{|S_t|} \frac{n_k}{n} w_t^k$
14:     sending $w_t$ to clients
15: end for
16: **Computed on clients:**
17: for $epoch \in Epochs$ do
18:     batches $\leftarrow$ partitioning data to B partitions
19:     for $b \in batches$ do
20:         $w \leftarrow w - \eta \nabla l(w; b; \omega_{(t-1)})$
21:     end for
22: end for
23: Sending weights to server.
24: Receiving averaged $w_t$
25: $E \leftarrow E \cup \{w_t\}$
26: Compute error $Error_t = \sum_{i=0}^{n} \omega_{i(t-1)} * I_{y_i \neq \hat{y}_i}$
27: Compute $\alpha_t = \frac{1}{2} log(\frac{1-Error_t}{Error_t})$.
28: Set $\omega_{it} \leftarrow \omega_{i(t-1)} * e^{-1^{I_{y_i \neq \hat{y}_i}} * \alpha_t}$, $i = 1, 2, \ldots, N$.
```

The FedMTBoost algorithm can be viewed from a global point, and this way it is essentially carrying out the same steps as FedAvg, but when it is viewed from the clients' local point it resembles the AdaBoost algorithm, with the managing of weights during training.

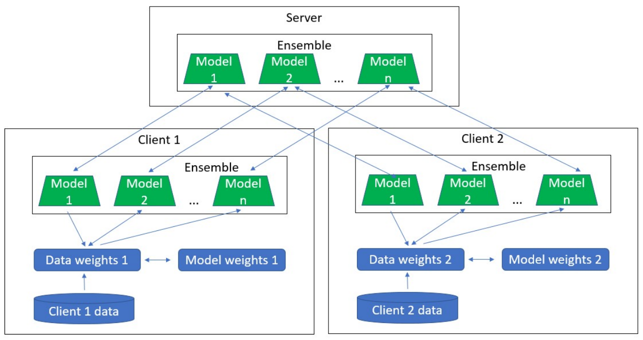


Fig. 1. High-level overview of FedMTBoost.

When predicting for new data, every client can use the shared ensemble and create a weighted average using their private model weights.

III. Results

To make sense of different technologies affecting the results of training and the methods' dependence on data we have devised a set of experiments to be evaluated on different dataset sizes. The experiments can be split along three axes: singletask and multitask, boosted and non-boosted methods, and finally based on the type of federation: singleparty, multiparty and federated. This resulted in the following experiments (with each having a single- and a multitask variant):

- Singleparty single Multilayer perceptron (MLP) model: All data pooled and one model trained
- Singleparty boosted: All data pooled and AdaBoost ensemble trained
- Multiparty single MLP model: Data distributed to 10 clients and everyone trains an MLP on their part of the data
- FedAvg: Data distributed and partners train MLP in a federated way, with the FedAvg algorithm
- FedMTBoost: Data distributed and clients train an ensemble with the FedMTBoost algorithm

A. Technical details

The methods were evaluated on the tyrosine kinase assays of the KIBA dataset, which resulted in 60 tasks unevenly distributed in the multiparty scenarios. The MLP was based on the SparseChem [16] implementation with one hidden layer with a size of 1400, using Adam optimizer. The federation of data happened both in the compound and the assay spaces, which means that averaging is only possible in the first layer, and not on the output (as different clients might use entirely different assays). We compare the AUROC scores of the methods on a five-fold dataset, with fold "0" always being used for evaluation. For folds 1, 2, 3 and 4, every possible subset is used for training, with the complementary folds being used for evaluation too.

B. Comparing singleparty methods

To see what effect boosting has on a baseline MLP method we first evaluated them on the whole dataset. The results below show the performances of boosting and MLP-based training, which as we can see in this scenario is not significant. The results are similar in a singletask training, but with worse performances.

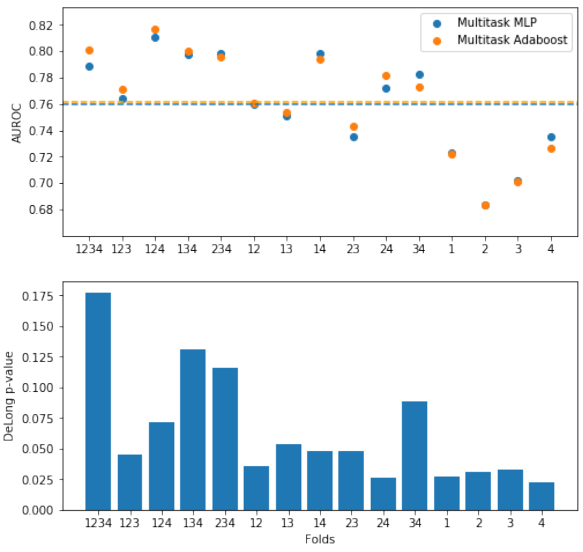


Fig. 2. AUROC scores and DeLong p-values of singleparty methods for every set of folds used in training.

TABLE I
Singleparty average AUROC scores for 4 fold trainings

	Singletask	*Multitask*
MLP	0.7249	0.7885
Boosting	0.7062	0.8007

C. Comparing multiparty methods

The more interesting results come, when the data is distributed and not every client has access to every assay and compound. First to get a baseline every client trains a model for themselves without federation, and this can be compared with the federated methods instead of a singleparty performance.

When looking at federated averaging we can see that for partners with enough data it does improve the predictive performance, which is expected as more information is present for these types of models, and they are able to develop a better representation. Although the improvement is present it is not statistically significant in most relevant cases. As a significance threshold of $p = 0.05$ would be required.

The case for FedMTBoost is different: It can improve the predictive performance of FedAvg and multiparty MLPs. The improvements here are in most cases significant and are present in both multitask and singletask scenarios. In Fig. 4

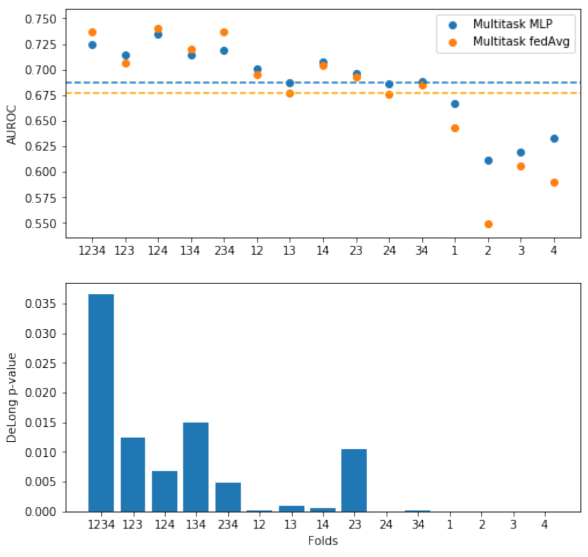


Fig. 3. AUROC scores and DeLong p-values of multiparty and federated methods for every set of folds used in training.

we can see the performance of partner 8, which is one of the larger partners, but it can demonstrate the usual scale of improvements.

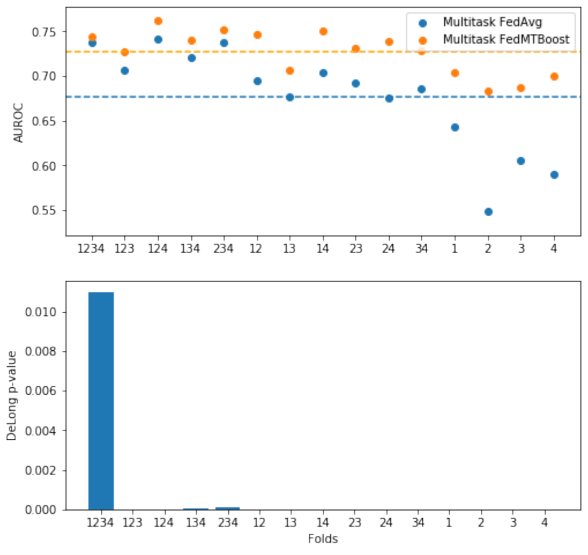


Fig. 4. AUROC scores and DeLong p-values of federated methods for every set of folds used in training.

It is important to note that the most significant differences were achieved when not all the data was present for training.

IV. Conclusion

As we showed Federated Boosting is a possible and useful approach that is adaptable to both single- and multitask environments. It does improve the predictive performance of FedAvg by leveraging data weighting from boosting. This proves that ensemble methods are a good fit to use in federated learning.

TABLE II
Multiparty average AUROC scores for 4 fold trainings on partner 8

	Singletask	*Multitask*
Multiparty MLP	0.6625	0.725
FedAvg	0.6235	0.7374
FedMTBoost	0.7325	0.7445

References

[1] Vamathevan, J., Clark, D., Czodrowski, P., Dunham, I., Ferran, E., Lee, G., ... & Zhao, S. (2019). Applications of machine learning in drug discovery and development. Nature reviews Drug discovery, 18(6), 463-477.

[2] Tamimi, N. A., & Ellis, P. (2009). Drug development: from concept to marketing!. Nephron Clinical Practice, 113(3), c125-c131.

[3] Chen, R., Liu, X., Jin, S., Lin, J., & Liu, J. (2018). Machine learning for drug-target interaction prediction. Molecules, 23(9), 2208.

[4] Sachdev, K., & Gupta, M. K. (2019). A comprehensive review of feature based methods for drug target interaction prediction. Journal of biomedical informatics, 93, 103159.

[5] Tang, J., Szwajda, A., Shakyawar, S., Xu, T., Hintsanen, P., Wennerberg, K., & Aittokallio, T. (2014). Making sense of large-scale kinase inhibitor bioactivity data sets: a comparative and integrative analysis. Journal of Chemical Information and Modeling, 54(3), 735-743.

[6] Gaulton, A., Bellis, L. J., Bento, A. P., Chambers, J., Davies, M., Hersey, A., ... & Overington, J. P. (2012). ChEMBL: a large-scale bioactivity database for drug discovery. Nucleic acids research, 40(D1), D1100-D1107.

[7] Sun, J., Jeliazkova, N., Chupakhin, V., Golib-Dzib, J. F., Engkvist, O., Carlsson, L., ... & Chen, H. (2017). ExCAPE-DB: an integrated large scale dataset facilitating Big Data analysis in chemogenomics. Journal of cheminformatics, 9(1), 1-9.

[8] McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017, April). Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics (pp. 1273-1282). PMLR.

[9] Caruana, R. (1997). Multitask learning. Machine learning, 28(1), 41-75.

[10] Xu, Y., Ma, J., Liaw, A., Sheridan, R. P., & Svetnik, V. (2017). Demystifying multitask deep neural networks for quantitative structure–activity relationships. Journal of chemical information and modeling, 57(10), 2490-2504.

[11] Svetnik, V., Wang, T., Tong, C., Liaw, A., Sheridan, R. P., & Song, Q. (2005). Boosting: An ensemble learning tool for compound classification and QSAR modeling. Journal of chemical information and modeling, 45(3), 786-799.

[12] Cheng, K., Fan, T., Jin, Y., Liu, Y., Chen, T., Papadopoulos, D., & Yang, Q. (2021). Secureboost: A lossless federated learning framework. IEEE Intelligent Systems, 36(6), 87-98.

[13] Li, Q., Wen, Z., & He, B. (2020, April). Practical federated gradient boosting decision trees. In Proceedings of the AAAI conference on artificial intelligence (Vol. 34, No. 04, pp. 4642-4649).

[14] Wang, B., & Pineau, J. (2015, February). Online boosting algorithms for anytime transfer and multitask learning. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 29, No. 1).

[15] Zhang, Y., & Yeung, D. Y. (2012, September). Multi-task boosting by exploiting task relationships. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 697-710). Springer, Berlin, Heidelberg.

[16] Arany, A., Simm, J., Oldenhof, M.,& Moreau, Y. (2022). SparseChem: Fast and accurate machine learning model for small molecules. arXiv preprint arXiv:2203.04676.