Demo Abstract: Event Stream Abstraction Using nfer

Sean Kauffman
Electrical & Computer Engineering
University of Waterloo
Waterloo N2L 3G1, Canada
skauffma@uwaterloo.ca

Sebastian Fischmeister
Electrical & Computer Engineering
University of Waterloo
Waterloo N2L 3G1, Canada
sfischme@uwaterloo.ca

ABSTRACT

We propose to demonstrate the open source implementation of `nfer` (http://nfer.io), a language and system for abstracting event streams. The tool is applicable to a wide variety of cyber-physical systems, and supports both manual and mined specifications. In addition to basic operation, we will demonstrate standalone monitor generation and integration with the popular R and Python languages. The demonstration will use real-world data captured from applications such as an autonomous vehicle and a hexacopter.

CCS CONCEPTS

• **Theory of computation** → **Rewrite systems**; • **Computer systems organization** → **Embedded software**; • **General and reference** → *Verification*;

KEYWORDS

telemetry comprehension, event stream processing, temporal logic, runtime verification

ACM Reference format:
Sean Kauffman and Sebastian Fischmeister. 2019. Demo Abstract: Event Stream Abstraction Using nfer. In *Proceedings of ICCPS '19: ACM/IEEE International Conference on Cyber-Physical Systems, Montreal, QC, Canada, April 16–18, 2019 (ICCPS '19),* 2 pages.
https://doi.org/10.1145/3302509.3313327

1 INTRODUCTION

In this demo, we propose to present the open source implementation of `nfer`. `Nfer` is a recently introduced formalism and system for machine and human comprehension of telemetry streams [3–5]. `Nfer` abstracts an event stream into a hierarchy of *intervals* related using a domain-specific language (DSL) based on Allen's Temporal Logic (ATL) [1]. The system has also been extended to mine these intervals from real-time system traces [2]. There are two, known implementations of `nfer`. One was written in Scala at the Jet Propulsion Laboratory (JPL), and another has been written in C and released under the GNU Public License version 3 (GPLv3) license at http://nfer.io. For the remainder of this document, `nfer` will refer to the C implementation.

`Nfer` has many potential uses for processing event streams and logs from cyber-physical systems. The tool was developed for abstracting spacecraft telemetry, but it is applicable to any event stream representing system behavior. For example, `nfer` is useful for processing Operating System (OS) system call logs. `Nfer` is useful for Runtime Verification (RV) of systems, but its abstractions also compress information. Furthermore, `nfer` consumes few system resources, allowing it to be run on a remote, embedded target.

In this demo we propose to demonstrate the operation of `nfer` using easy to understand examples. We will cover simple concepts from the `nfer` language where it is necessary to understand the operation of the tool. The tool can be run directly using a supplied specification, or a standalone monitor can be generated for use on a real-time system. The tool can also mine specifications, given a real-time system trace. We will also demonstrate the tool's integration with the R and Python languages.

2 CONCEPTS TO DEMONSTRATE

We propose to demonstrate the following concrete features of the `nfer` tool. By the end of the demonstration the audience should feel comfortable performing any of these tasks.

- **Abstracting a trace with `nfer`:**
  We will show a simple specification for abstracting Controller Area Network (CAN) logs from an autonomous vehicle. The logs are from a real-world example and show how `nfer` can be used to solve problems in practice.
- **Mining a specification with `nfer`:**
  We will demonstrate mining a specification from a QNX system trace using `nfer`. The mined specification can then be applied to extract interval abstractions from the same or other traces.
- **Building a runtime monitor with `nfer`:**
  `Nfer` support real-time embedded environments by producing standalone monitors without system requirements like dynamic memory. We will show how to produce and run such a monitor for a small embedded system.
- **Using `nfer` from R:**
  We will demonstrate the use of `nfer`'s R Application Programming Interface (API). R is a popular language for statistical processing and `nfer` can be useful in this environment to help abstract large datasets.
- **Using `nfer` from Python:**

ICCPS '19, April 16–18, 2019, Montreal, QC, Canada

ACM ISBN 978-1-4503-6285-6/19/04…$15.00
https://doi.org/10.1145/3302509.3313327

We will also show `nfer`'s Python API. The Python API is geared toward application developers who want to include `nfer` in their projects.

3 CONCLUSION

`Nfer` is a valuable tool about which cyber-physical systems researchers and practitioners should want to learn. We propose to demonstrate the tool, which is available under the GPLv3 license. By the end of the demonstration, the audience should feel comfortable installing and using `nfer` in a variety of contexts.

We propose to demonstrate several abilities of the `nfer` tool. We will show how to apply a manually written specification to abstract an event stream, and how to mine a specification from a trace. We will demonstrate how to generate a standalone monitor using `nfer` for use in real-time and resource constrained environments. Finally, we will demonstrate `nfer`'s R and Python APIs.

REFERENCES

[1] James F Allen. 1983. Maintaining knowledge about temporal intervals. *Commun. ACM* 26, 11 (1983), 832–843.

[2] Sean Kauffman and Sebastian Fischmeister. 2017. Mining Temporal Intervals from Real-time System Traces. In *Proceedings of the 6th International Workshop on Software Mining (SoftwareMining)*. Champaign, USA, 1–8. https://doi.org/10.1109/SOFTWAREMINING.2017.8100847

[3] Sean Kauffman, Klaus Havelund, and Rajeev Joshi. 2016. nfer–A Notation and System for Inferring Event Stream Abstractions. In *International Conference on Runtime Verification*. Springer, 235–250. https://doi.org/10.1007/978-3-319-46982-9_15

[4] Sean Kauffman, Klaus Havelund, Rajeev Joshi, and Sebastian Fischmeister. 2018. Inferring Event Stream Abstractions. *Formal Methods in System Design* 53, 1 (01 08 2018), 54–82. https://doi.org/10.1007/s10703-018-0317-z

[5] Sean Kauffman, Rajeev Joshi, and Klaus Havelund. 2016. Towards a logic for inferring properties of event streams. In *International Symposium on Leveraging Applications of Formal Methods*. Springer, 394–399. https://doi.org/10.1007/978-3-319-47169-3_31