Encoding Forensic Multimedia Evidence from MARF Applications as Forensic Lucid Expressions

Serguei A. Mokhov
SGW, EV7.139-2
Department of Computer Science and Software Engineering
Concordia University, Montreal, Quebec, Canada
Email: mokhov@cse.concordia.ca

***Abstract*—In this work we summarize biometric evidence as well as file type evidence extraction "exported" as formal Forensic Lucid language expression in the form of higher-order intensional contexts for further case analysis by a system that interprets Forensic Lucid expressions for claim verification and event reconstruction. The digital evidence is exported from the Modular Audio Recognition Framework (MARF)'s applications runs on a set of data comprising biometric voice recordings for speaker, gender, spoken accent, etc. as well as more general file type analysis using signal and pattern recognition processing techniques. The focus is in translation aspect of the extracted evidence into formal Forensic Lucid expressions for further analysis.**

***Index Terms*—encoding multimedia evidence, evidence analysis, forensic case specification, Forensic Lucid, higher-order intensional contexts, Lucid, MARFL, Modular Audio Recognition Framework (MARF)**

I. Introduction

Problem Statement: Authors of MARF [1] proposed to use some of its applications for biometric forensic analysis of multimedia data such as audio recordings [2] and scanned handwritten images [3] as well as file-type analysis [4]. On the other hand, they proposed an intensional scripting language for dynamic configuration management and scripting of the said applications, MARFL [5]. Yet they provide no convenient means to extract and encode the identified evidence for further processing and case analysis. In our research, we work with the Forensic Lucid specification language and need to be able to encode the extracted information as contextual expressions syntactically and semantically valid in Forensic Lucid.

Proposed Solution: We propose an adapter translator program, that translates the MARF's data structures in a way similar to the way MARFL represents configuration details, but translated into the Forensic Lucid-compatible definitions. We add it as back-end plug-in attached to MARF to compile the resulting data structures into the Forensic Lucid expressions.

A. Forensic Lucid

Forensic Lucid [6], [7] is a forensic case specification language based on the intensional logic and programming paradigms [8], [9], [10], [11], [12], [13], [14], [15]. The authors of Forensic Lucid formally design and specify the language to be able to "script" in a tool the evidence, stories told by witnesses, and the case itself as an evidential statement as a higher-order context specification (all Lucid dialects are context-oriented with the contexts being first-class values) as well as the case specification in terms of inference derivations of the case. Forensic Lucid was created to address the difficulties and complexity to use the earlier formal approach [16], [17] that required a potential investigator to construct a finite-state machine (FSM) and model transitions. Forensic Lucid inherited some of the terminology and formalities in the specification of events, properties, observations, observation sequences, evidential statement, and the transition function and its inverse for event reconstruction and automated verifiability of traces from the FSM approach. Forensic Lucid's compiler and development environment is being realized in a sister project, the General Intensional Programming System (GIPSY) [13], [18], [19].

B. MARF and MARFL

MARFL is an intensional Lucid-like configuration specification language for MARF applications scripting support. It supports higher-order context definitions for nested configuration parameters. We capitalize on that notion in our research by translating the results of classification and parts of the configurations into the Forensic Lucid expressions. A comprehensive example of a context for processing a WAV file in the MARF's pipeline can be modeled as shown in Figure 1 [5]. In that example the authors of MARFL illustrate a complex hierarchical context expression where several nested dimensions are explicitly specified. In general, MARFL's context follows the classical Lucid definition, where it is defined as a collection of $<dimension : tag>$ pairs. What MARFL does differently is that a single pair may not necessarily be an atomic context, but may contain sub-contextual elements. The inner-most context is always simple and atomic and typically has dimensions of primitive types, such as integer, IEEE 754 floating point value, or a string. The outer layers of context hierarchy are composite objects. Thus, a `[sample loader:WAV]` denotes a dimension of type `sample loader` with its higher-order tag value `WAV`. The `WAV` dimension value can further be decomposed to an atomic simple context if needed, that contains three dimensions of primitive types.

Such a way of context representation of the higher-order context by the MARFL authors is similar to equivalent definitions described by Swoboda et al. in [20], [21], [22], [23], where a tree of contexts is defined for languages like iHTML (with nested tags), functional intensional databases annotated with XML, etc.

II. Methodology

A. Translation into Forensic Lucid

The higher-order contextual specification of Forensic Lucid goes (top to bottom) from `evidential statement`, to `observation sequence`, to `observation`, to $(P, \mathit{min}, \mathit{max})$, where the observed property P is an arbitrary object, usually a human-readable description of the state or event, and $[\mathit{min}, \mathit{min}{+}\mathit{max}]$ is the duration of the observation of that property [16], [17]. Thus, at the higher order we need something similar to Listing 1's `o1`. We borrow the mapping of the properties P (that can be strings, integers, or even any-order contextual or otherwise expressions) to a human-readable rewrite string, similarly to what is in Listing 2 after the => sign. We also assume that a P by itself is a syntactic sugar of $(P, 1, 0)$.

```
evidential statement es = unordered {os1,os2,os3};
where
  observation sequence os1 = ordered {o1,o2,o3};
  observation sequence os2 = ordered {o4,o5,o6};
  observation sequence os3 = ordered {o7,o8,o9};
  where
    observation o1 =
    [
      p:‘‘ID:23:JoeAverage’’ => ‘‘speaker ID is 23, Joe
          Average’’,
      min:1,
      max:0
    ];
    ...
  end;
end;
```

Listing 1. Forensic Lucid Contextual Expression

B. MARF Evidence

The evidence extracted from the analysis results of MARF comes from the several internal data structures, namely `Result`, `ResultSet`, `TrainingSet`, and `Configuration`.

The `Result` consists of tuples containing ID and *outcome*, which are the properties of a single result. The result set, `ResultSet`, is a collection of such tuples. Processed utterances (a.k.a feature vectors or clusters of `doubles`), alongside with the training file names and IDs comprise the training set data, and configuration is a collection of processing settings that led to the current results given the training set.

We need to specify what is the property P in the three categories (configuration, training set, and the result set) and what is its observed duration. We set it as follows: it is a default of $(1, 0)$, as the notion of duration varies per configuration, so we are only interested in of how we arrive from the given configuration and training set to the results.

We syntactically write `observation o = P`, which is equivalent to $(P, 1, 0)$ as mentioned earlier. We need to then determine what is an observation sequence. We define it as a sequence of three “observations”, each observation per category. The observations must be ordered: (1) configuration `configo`, (2) training set `tseto`, and (3) the classification result `resulto`. The meaning of this observation sequence is that given some MARF configuration settings and the existing training set, the system produces the classification result. If we are performing the training, the observation sequence is slightly different, but also has three observations: configuration, incoming sample, and the resulting training set, which would be encoded accordingly as all the necessary primitives for that are already defined. With such notions in mind we come up with the complete exportable Forensic Lucid expression as a 3-observation sequence, e.g. presented in Listing 2. As with our simplifying assumption, we can remove the $(1, 0)$ syntactical constructs, and just keep the P, which in this case is a higher-order context specification, as shown in Figure 3.

```
MrA @ es_mra
where
  evidential statement es_mra = {os_mra, os_final,
      os_unrelated};

  observation sequence os_mra = ($, o_unrelated_clean, $,
      o_blackmail, $);
  observation sequence os_final = ($, o_final);
  observation sequence os_unrelated = ($, o_unrelated, $, (
      Ct,0,0), $);

  observation o_final = (1, ”u”, ”t2”);
  observation o_unrelated_clean = (1, ”u”, ”o1”);

  // ...

  invtrans(Q, es_mra, o_final) = backraces
  where
    // list of all possible dimensions
    observation Q = lengths box left_part box right_part;

    // events
    observation lengths = unordered {0, 1, 2};

    // symbolic labels map to human descriptions
    observation left_part = unordered {
      ”u”  => ”unrelated”,
      ”t1” => ”threats−obscured part”,
      ”o1” => ”other data (left part)”
    };

    observation right_part = unordered {
      ”t2” => ”threats in slack”,
      ”o2” => ”other data (right part)”
    };

    backtraces = [ A, B, C, D, ];
    where
    ...
    end;
  end;
end;
```

Listing 2. Blackmail Case Modeling in Forensic Lucid

We define some syntactic examples and the corresponding simplifications to illustrate a few points:

- an observation sequence OS is a sequence of three observations, where the last one is the “no-observation” $ construct:

  `os = { o1, o2, $ };`

```
[
  sample loader      : WAV [ channels: 2, bitrate: 16, encoding: PCM, f : 8000 ],
  preprocessing      : LOW-PASS-FFT-FILTER [ cutoff: 2024, windowsize: 1024 ],
  feature extraction : LPC [ poles: 20, windowsize: 1024 ],
  classification     : MINKOWSKI-DISTANCE [ r : 5 ]
]
```

Fig. 1. Example of hierarchical context specification for a evaluation configuration of MARF.

```
MARFos = { confo, tseto, resulto } =
{
  ([
    sample loader      : WAV [ channels: 2, bitrate: 16, encoding: PCM, f : 8000 ],
    preprocessing      : LOW-PASS-FFT-FILTER [ cutoff: 2024, windowsize: 1024 ],
    feature extraction : LPC [ poles: 20, windowsize: 1024 ],
    classification     : MINKOWSKI-DISTANCE [ r : 5 ]
  ], 1, 0),

  ([data:{[5.2,3.5,7.5],[3.6,2.5,5.5,6.5]}, files:[``/foo/bar.wav'',``/bar/foo.wav'']], 1, 0),

  ([ID:5, outcome:1.5], 1, 0)
}
```

Fig. 2. Example of a three-observation sequence context exported from MARF to Forensic Lucid.

```
MARFos = { confo, tseto, resulto } =
{
  [
    sample loader      : WAV [ channels: 2, bitrate: 16, encoding: PCM, f : 8000 ],
    preprocessing      : LOW-PASS-FFT-FILTER [ cutoff: 2024, windowsize: 1024 ],
    feature extraction : LPC [ poles: 20, windowsize: 1024 ],
    classification     : MINKOWSKI-DISTANCE [ r : 5 ]
  ],

  [data:{[5.2,3.5,7.5],[3.6,2.5,5.5,6.5]}, files:[``/foo/bar.wav'',``/bar/foo.wav'']],

  [ID:5, outcome:1.5]
}
```

Fig. 3. Example of a simplified three-observation sequence context exported from MARF to Forensic Lucid.

- if observations of properties $P_1 = [a:1, b:2]$ and $P_2 = [s:4, g:7]$ (which happened to be contexts themselves here) have a duration of 1 (one) they can be shortened in their expression to just P. E.g. the following observation sequences are equivalent:

  ```
  os = {[a:1,b:2], [s:4,g:7]};
  os = {([a:1,b:2],1,0), ([s:4,g:7],1,0)};
  ```

- in generic observation sequences where min and max duration parameters are not zero, cannot be implicit. In the below is an example of a "complex" observation sequence where P_1 and P_2 have several possible durations, e.g. in $os - \{o_1, o_2\}$, where $o_1 = (P_1, 5, 4)$ and $o_2 = (P_2, 1, 2)$, would result in:

  ```
  os = {([a:1,b:2],5,4), ([s:4,g:7],1,2)};
  ```

- As a "syntactical sugar" we allow a declaration of an observation sequence os that consists of only a single observation o_1, we allow dropping of the curly braces:

  ```
  os = o1; <=> os = {o1};
  ```

III. Limitations

There are a number of current limitations with the approach that are to be addressed in the upcoming future work. We list some of them here:

- At this point the investigator will have to "copy-paste" the produced output into their Forensic Lucid case specification for further evaluation after the output is produced, i.e. there are now friendly user-interface or any other type of integration of the exporter code, MARF, and GIPSY.
- The concrete syntax and semantics of Forensic Lucid and MARFL, are not fully finalized as of this writing as they both go through the design, formalization, and analysis phases with relatively frequent adjustments as the research moves forward.
- There are no correctness proofs yet for the Forensic Lucid code itself as well as the correctness of implementation

- of MARF and GIPSY, which are necessary to be valid for the tool to be usable in court.

IV. Conclusion

We devised a basic methodology of exporting and translating the evidence contained within MARF's data structures represented in the higher level in the MARFL language, as a collection of contextual expressions in Forensic Lucid. The evidence of biometric origin, file type analysis, writer analysis and others can therefore be exported into Forensic Lucid for case formulation later on. An investigator can simply use the provided expression in their Forensic Lucid case as-is to maintain the library of observation sequences with the collected evidence.

V. Future Work

This sections lists a number of items to improve in the near future work in the Forensic Lucid language and the surrounding systems. These are mostly there to address the limitations described earlier and enhance overall usability, applicability, and standardization of the language:

- Prove the correctness of the MARF code and its storage modules in the internal representation of the evidence.
- Complete formal definition of Forensic Lucid and MARFL, and then formally verify the correctness of the adapter/exporter code and prove its equivalence between the MARFL and Forensic Lucid representations in Isabelle [24].
- Provide equivalent translation and export tools for the JPF-based forensic toolkit [25], [26], [27] plug-ins for memory, log, and email analysis evidence as evidential expressions specified in Forensic Lucid.
- Export MARF pipeline as a transition function ψ for complete specification of the MARF-based system in Forensic Lucid.

Acknowledgment

The author thanks Drs. Joey Paquet and Mourad Debbabi for the helpful and detailed review, suggestions, support, and comments about this work. We acknowledge reviewers who took time to review this work and provide us with constructive feedback. This work is sponsored in part by the Faculty of Engineering and Computer Science, Concordia University, Montreal, Canada.

References

[1] The MARF Research and Development Group, "The Modular Audio Recognition Framework and its Applications," SourceForge.net, 2002–2008, http://marf.sf.net, last viewed December 2008.

[2] S. A. Mokhov, "Study of best algorithm combinations for speech processing tasks in machine learning using median vs. mean clusters in MARF," in *Proceedings of C3S2E'08*, B. C. Desai, Ed. Montreal, Quebec, Canada: ACM and BytePress, May 2008, pp. 29–43, ISBN 978-1-60558-101-9.

[3] ——, "Writer Identification Using Inexpensive Signal Processing Techniques: Experimental Results," 2008, unpublished.

[4] S. A. Mokhov and M. Debbabi, "File type analysis using signal processing techniques and machine learning vs. `file` unix utility for forensic analysis," in *Proceedings of the IT Incident Management and IT Forensics (IMF'08)*, O. Goebel, S. Frings, D. Guenther, J. Nedon, and D. Schadt, Eds., Mannheim, Germany, Sep. 2008, pp. 73–85, LNI140.

[5] S. A. Mokhov, "Towards syntax and semantics of hierarchical contexts in multimedia processing applications using MARFL," in *Proceedings of the 32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC)*. Turku, Finland: IEEE Computer Society, Jul. 2008, pp. 1288–1294.

[6] S. A. Mokhov and J. Paquet, "Formally specifying and proving operational aspects of Forensic Lucid in Isabelle," Department of Electrical and Computer Engineering, Concordia University, Tech. Rep. 2008-1-Ait Mohamed, Aug. 2008, in Theorem Proving in Higher Order Logics (TPHOLs2008): Emerging Trends Proceedings.

[7] S. A. Mokhov, J. Paquet, and M. Debbabi, "Formally specifying operational semantics and language constructs of Forensic Lucid," in *Proceedings of the IT Incident Management and IT Forensics (IMF'08)*, O. Goebel, S. Frings, D. Guenther, J. Nedon, and D. Schadt, Eds., Mannheim, Germany, Sep. 2008, pp. 197–216, LNI140.

[8] E. A. Ashcroft and W. W. Wadge, "Lucid - a formal system for writing and proving programs," *SIAM J. Comput.*, vol. 5, no. 3, 1976.

[9] ——, "Erratum: Lucid - a formal system for writing and proving programs." *SIAM J. Comput.*, vol. 6, no. (1):200, 1977.

[10] ——, "Lucid, a nonprocedural language with iteration," *Communication of the ACM*, vol. 20, no. 7, pp. 519–526, Jul. 1977.

[11] W. Wadge and E. Ashcroft, *Lucid, the Dataflow Programming Language*. London: Academic Press, 1985.

[12] E. Ashcroft, A. Faustini, R. Jagannathan, and W. Wadge, *Multidimensional, Declarative Programming*. London: Oxford University Press, 1995.

[13] J. Paquet, "Scientific intensional programming," Ph.D. dissertation, Department of Computer Science, Laval University, Sainte-Foy, Canada, 1999.

[14] R. Lalement, *Computation as Logic*. Prentice Hall, 1993, C.A.R. Hoare Series Editor. English translation from French by John Plaice.

[15] P. Rondogiannis, "Higher-order functional languages and intensional logic," Ph.D. dissertation, Department of Computer Science, University of Victoria, Victoria, Canada, 1994.

[16] P. Gladyshev, "Finite state machine analysis of a blackmail investigation," in *International Journal of Digital Evidence*. Technical and Security Risk Services, Sprint 2005, Volume 4, Issue 1, 2005.

[17] P. Gladyshev and A. Patel, "Finite state machine approach to digital event reconstruction," in *Digital Investigation Journal*, vol. 2, 2004.

[18] J. Paquet and P. Kropf, "The GIPSY architecture," in *Proceedings of Distributed Computing on the Web*, Quebec City, Canada, 2000.

[19] J. Paquet, "A multi-tier architecture for the distributed eductive execution of hybrid intensional programs," 2008, submitted for publication at SAC'09.

[20] P. Swoboda, "A formalisation and implementation of distributed intensional programming," Ph.D. dissertation, The University of New South Wales, Sydney, Australia, 2004.

[21] P. Swoboda and W. W. Wadge, "Vmake, ISE, and IRCS: General tools for the intensionalization of software systems," in *Intensional Programming II*, M. Gergatsoulis and P. Rondogiannis, Eds. World-Scientific, 2000.

[22] P. Swoboda and J. Plaice, "A new approach to distributed context-aware computing," in *Advances in Pervasive Computing*, A. Ferscha, H. Hoertner, and G. Kotsis, Eds. Austrian Computer Society, 2004, ISBN 3-85403-176-9.

[23] ——, "An active functional intensional database," in *Advances in Pervasive Computing*, F. Galindo, Ed. Springer, 2004, pp. 56–65, LNCS 3180.

[24] L. C. Paulson and T. Nipkow, "Isabelle: A generic proof assistant," University of Cambridge and Technical University of Munich, 2007, http://isabelle.in.tum.de/, last viewed: December 2007.

[25] M. Debbabi, A. R. Arasteh, A. Sakha, M. Saleh, and A. Fry, "A collection of JPF forensic plug-ins," Computer Security Laboratory, Concordia Institute for Information Systems Engineering, 2007–2008.

[26] A. R. Arasteh and M. Debbabi, "Forensic memory analysis: From stack and code to execution history," *Digital Investigation Journal*, vol. 4, no. 1, pp. 114–125, Sep. 2007.

[27] A. R. Arasteh, M. Debbabi, A. Sakha, and M. Saleh, "Analyzing multiple logs for forensic evidence," *Digital Investigation Journal*, vol. 4, no. 1, pp. 82–91, Sep. 2007.