arXiv:1107.0940v2 [cs.PL] 5 Jul 2013

Furthering Baseline Core Lucid Standard Specification in the Context of the History of Lucid, Intensional Programming, and Context-Aware Computing

Paquet, Joey
Mokhov, Serguei A.

SIGLUCID

Abstract

This work is multifold. We review the historical literature on the Lucid programming language, its dialects, intensional logic, intensional programming, the implementing systems, and context-oriented and context-aware computing and so on that provide a contextual framework for the converging Core Lucid standard programming model. We are designing a standard specification of a baseline Lucid virtual machine for generic execution of Lucid programs. The resulting Core Lucid language would inherit the properties of generalization attempts of GIPL (1999–2013) and TransLucid (2008–2013) for all future and recent Lucid-implementing systems to follow. We also maintain this work across local research group in order to foster deeper collaboration, maintain a list of recent and historical bibliography and a reference manual and reading list for students. We form a (for now informal) SIGLUCID group to keep track of this standard and historical records with eventual long-term goal through iterative revisions for this work to become a book or an encyclopedia of the referenced topics, and perhaps, an RFC. We first begin small with this initial set of notes.

Contents

1 Introduction **2**
1.1 Motivation . . . 3
1.2 Proposed Solution . . . 3
1.3 SIGLUCID . . . 3

2 Historical Perspective, Context, Dialects, and Applications **3**
2.1 Lucid Dialects . . . 4
2.1.1 Incomplete Brief History and The Family . . . 5
2.2 List of Tools and Implementing Systems . . . 6
2.3 Application Domains . . . 6
2.4 Related Work . . . 7

3 Core Lucid Standard Specification Design **19**
3.1 SIGLUCID Meetings . . . 19
3.1.1 SECASA 2010 Meeting at SERA 2010, Montreal, Canada . . . 19
3.1.2 SECASA 2009 Meeting at COMPSAC 2009, Seattle, USA . . . 19
3.1.3 SECASA 2008 Meeting at COMPSAC 2008, Turku, Finland . . . 20
3.1.4 PLC 2005 Meeting at WORLDCOMP 2005, Las Vegas, USA . . . 22
3.2 TransLucid . . . 23
3.3 GIPSY . . . 23

3.3.1 Hybrid Interaction with Other Languages 23
3.3.2 Introduction to the GIPSY Type System 26
3.4 The Core Lucid Standard 30
3.4.1 Syntax 30
3.4.2 Semantics 30

4 Conclusion **31**

5 Future Work **31**

6 Acknowledgments **31**

List of Figures

1 GIPC Framework with the Preprocessor 24
2 Example of Eductive Evaluation of Objective Lucid Progran 25
3 GIPSY Type System. 29

Listings

1 Example of a hybrid GIPSY program. 27

List of Tables

1 Matching data types between Lucid and Java. 28

1 Introduction

This work gears toward a generalization on a number of previous results by various authors in terms of context specification in the Lucid programming language and the Core Lucid dealing with data types and have a virtual machine standard agreed to by SIGLUCID. Aside from various data types (primarily to address the hybrid computing paradigms uniformly of Lucid integrating with imperative dialects), the context definition should also be hierarchical for certain application domains to allow for context nesting. The notion of context is central to Lucid as an explicit meaning component that is specified as a first class-value. Traditional Lucid's context specification was assuming tags and the corresponding values were *simple* – i.e. a collection of dimension names and the value pairs would denote a point in the context space. Then, the notion of point was not sufficient for some Lucid dialects that needed higher-order contextual notions, such as context sets to denote a context area or field instead of a point, as it was done in Lucx. Another way to traverse a more complex notion of the context definition was done in iHTML and related tools where nesting of the tags would denote the nesting of contextual expressions forming a sort of contextual tree, where the actual tag values were at the leaves of the tree. Then, a similar need arose in Forensic Lucid and MARFL to specify higher-order contexts representing evidence and witness stories or configuration details, but allowing evaluation at any level of the context tree rather than just the leaves. Thus, this work aims at unifying and standardizing various context specifications under one uniform intermediate form that all Lucid dialects can adhere to thereby making the community speak the same language and potentially

bring interoperability between various Lucid implementations and incarnations across University groups working in the intensional programming domain.

1.1 Motivation

Higher-order context specification is needed for nested-level context that traditionally decomposes a higher-order value into its components, and equivalently from the components get to the parent component. This is partitioned in any nested markup-like language, e.g. iHTML, any XML-based definitions and descriptions of data and databases, configuration management of a software system components, as well as domain-specific applications such as contextual specification of a cyberforensic case where evidential statement is comprised of observation sequences representing encoded stories told by evidence and witnesses, which in turn decompose into observations, and then into properties and duration components; which all-in-all comprise a context of evaluation of a cyberforensic case. Thus, the need for higher-order contexts is apparent as a fundamental pillar supporting higher-order intensional logic (HOIL).

Types other than the context also should be exposed to the programmer when needed and allow for a wider range of data types and type systems to allow hybrid dialect interaction easier as well as compiler optimization and run-time system parallelizations.

1.2 Proposed Solution

For the context specification, we propose to extend the notion of context to be a bi-directional tree with the operators from GIPL, Lucx, iHTML and MARFL to query, switch, and traverse the depth of the context hierarchy. The language that encompasses the new specification on the syntax and semantic level is proposed to be called *Core Lucid* or *Standard Lucid* or *Nominal Lucid*.

The type specification and code segments are augmented as presented in possible specification from the SIGLUCID meetings and others in Section 3.1 and Section 3.3.1.

1.3 SIGLUCID

SIGLUCID: Special Interest Group on Lucid, Ubiquity, Context, Intensionality, and multi-Dimensionality. SIGLUCID is a working group of researchers in Lucid, intensional programming, intensional logic, context-aware and context-oriented computing and the related application domains (see Figure 2.3.
SIGLUCID currently is a loose affiliation of researchers, collaborators, and supporters in intensional logic, intensional programming, context-aware computing, etc. across Canada, Australia, and other places.

Should you wish to be a part and contribute, contact the people listed at the title page. This is a running draft to fill in the missing information as it becomes possible.

2 Historical Perspective, Context, Dialects, and Applications

The history of Lucid, multi-dimensional intensional programming and logic, context-orientation, parallel, concurrent, and distributed eductive evaluation aspects can be traced through different Lucid dialects, outlined in Section 2.1.

2.1 Lucid Dialects

Here we enumerate the Lucid dialects that came to be from either practical implementations and/or theoretical frameworks to study the intensionality properties, context, and mathematical and intensional logic foundations. We plan to make the list into a table or other presentation means with the status of each language and the related citations.

- Lucid
- GIPL
- TransLucid
- Lucx
- GLU
- GLU#
- Indexical Lucid
- Tensor Lucid
- Partial Lucid
- JLucid
- Objective Lucid
- Onyx
- Forensic Lucid
- JOOIP
- MARFL
- IHTML
- IHTML2
- iPerl
- ISE
- vmake
- Lustre
- pLucid
- Luthid

2.1.1 Incomplete Brief History and The Family

From 1974 to Lucid Today (taken from [59], incomplete, to be updated:

1. Lucid as a Pipelined Dataflow Language through 1974-1977. Lucid was introduced by Anchroft and Wadge in [7, 8]. Features:
   - A purely declarative language for natural expression of iterative algorithms.
   - Goals: semantics and verification of correctness of programming languages (for details see [7, 8]).
   - Operators as pipelined streams: one for initial element, and then all for the successor ones.
2. Intensions, Indexical Lucid, GRanular Lucid (GLU, [41, 42]), circa 1996. More details on these two dialects are provided further in the chapter as they directly relate to the theme of this thesis. Features:
   - Random access to streams in Indexical Lucid.
   - First working hybrid intensional-imperative paradigm (C/Fortran and Indexical Lucid) in the form of GLU.
   - Eduction or demand-driven execution (in GLU).
3. Partial Lucid, Tensor Lucid, 1999 [86].
   - Partial Lucid is an intermediate experimental language used for demonstrative purposes in presenting the semantics of Lucid in [86].
   - Tensor Lucid dialect was developed by Joey Paquet for plasma physics computations to illustrate advantages and expressiveness of Lucid over an equivalent solution written in Fortran.
4. GIPL, 1999 [86].
   - All Lucid dialects can be translated into this basic form of Lucid, GIPL through a set of translation rules. (GIPL is in the foundation of the execution semantics of GIPSY and its GIPC and GEE because its AST is the only type of AST GEE understands when executing a GIPSY program).
5. RLucid, 1999, [31]
   - A Lucid dialect for reactive real-time intensional programming.
6. JLucid, Objective Lucid, 2003 - 2005
   - These dialects introduce a notion of hybrid and object-oriented programming in the GIPSY with Java and Indexical Lucid and GIPL, and are discussed great detail in the follow up chapters of this thesis.
7. Lucx [139], 2003 - 2005
   - Kaiyu Wan introduces a notion of contexts as first-class values in Lucid, thereby making Lucx the true intensional language.

8. Onyx [35], April 2004.
    - Peter Grogono makes an experimental derivative of Lucid – Onyx to investigate on lazy evaluation of arrays.
9. GLU# [84], 2004
    - GLU# is an evolution of GLU where Lucid is embedded into C++.

2.2 List of Tools and Implementing Systems

1. GIPSY [93]
2. GLU
3. TransLucid
4. pLucid
5. `libintense`

2.3 Application Domains

1. Context-Aware Computing
2. Scientific Computing
3. Distributed and Parallel Evaluation
4. Ubiquitous and Mobile Computing
5. Wiki
6. Forensic Computing
7. Multimedia and Configuration Management
8. Program Verification
9. Software Engineering
10. Aspect-Oriented Programming
11. Web OS
12. Reactive Computing
13. Pervasive Computing
14. Autonomic Computing
15. Modeling and Simulation
16. Model Checking

2.4 Related Work

There is a vast amount of related and past work done. Over time we will provide brief historical description of each or a group of works clustered by a specific theme either in this section or relevant other sections. For now, however, we begin by citing them first, so anyone looking for the references can look them up in a jiffy and make their choice accordingly. This is ideal for graduate students and researchers starting in the subjects or looking for what's been done that they can benefit from.

Most recent on top:

- 2012
  - Sleiman Rabah, Serguei A. Mokhov, and Joey Paquet. An interactive graph-based automation assistant: A case study to manage the GIPSY's distributed multi-tier runtime system. [online], June 2012. Submitted for publication to RACS'13; Pre-print: `http://arxiv.org/abs/1212.4123`
- 2011
  - John Plaice. Cartesian programming. Technical Report UNSW-CSE-TR-1101, University of Grenoble, France, January 2011. Habilitation Thesis, online at `ftp://ftp.cse.unsw.edu.au/pub/doc/papers/UNSW/1101.pdf`
  - Yi Ji. Scalability evaluation of the GIPSY runtime system. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, March 2011
  - Serguei A. Mokhov, Joey Paquet, and Mourad Debbabi. Reasoning about a simulated printer case investigation with Forensic Lucid. In Pavel Gladyshev and Marcus K. Rogers, editors, *Proceedings of ICDF2C'11*, number 0088 in LNICST, pages 282–296. Springer, October 2011. ISBN 978-3-642-35514-1. doi: 10.1007/978-3-642-35515-8_23. Submitted in 2011, appeared in 2012; online at `http://arxiv.org/abs/0906.5181`
  - Serguei A. Mokhov, Joey Paquet, and Mourad Debbabi. On the need for data flow graph visualization of Forensic Lucid programs and forensic evidence, and their evaluation by GIPSY. In *Proceedings of the Ninth Annual International Conference on Privacy, Security and Trust (PST), 2011*, pages 120–123. IEEE Computer Society, July 2011. ISBN 978-1-4577-0582-3. doi: 10.1109/PST.2011.5971973. Short paper; full version online at `http://arxiv.org/abs/1009.5423`
- 2010
  - Serguei A. Mokhov and Joey Paquet. A type system for higher-order intensional logic support for variable bindings in hybrid intensional-imperative programs in GIPSY. In Tokuro Matsuo, Naohiro Ishii, and Roger Lee, editors, *9th IEEE/ACIS International Conference on Computer and Information Science, IEEE/ACIS ICIS 2010*, pages 921–928. IEEE Computer Society, May 2010. ISBN 978-0-7695-4147-1. doi: 10.1109/ICIS.2010.156. Presented at SERA 2010; online at `http://arxiv.org/abs/0906.3919`
  - Serguei A. Mokhov and Joey Paquet. Using the General Intensional Programming System (GIPSY) for evaluation of higher-order intensional logic (HOIL) expressions.

In *Proceedings of SERA 2010*, pages 101–109. IEEE Computer Society, May 2010. ISBN 978-0-7695-4075-7. doi: 10.1109/SERA.2010.23. Online at `http://arxiv.org/abs/0906.3911`

- Aihua Wu, Joey Paquet, and Serguei A. Mokhov. Object-oriented intensional programming: Intensional Java/Lucid classes. In *Proceedings of SERA 2010*, pages 158–167. IEEE Computer Society, 2010. ISBN 978-0-7695-4075-7. doi: 10.1109/SERA.2010.29. Online at: `http://arxiv.org/abs/0909.0764`
- Bin Han, Serguei A. Mokhov, and Joey Paquet. Advances in the design and implementation of a multi-tier architecture in the GIPSY environment with Java. In *Proceedings of SERA 2010*, pages 259–266. IEEE Computer Society, 2010. ISBN 978-0-7695-4075-7. doi: 10.1109/SERA.2010.40. Online at `http://arxiv.org/abs/0906.4837`
- Bin Han. Towards a multi-tier runtime system for GIPSY. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, 2010
- Yi Ji, Serguei A. Mokhov, and Joey Paquet. Unifying and refactoring DMF to support concurrent Jini and JMS DMS in GIPSY. In Bipin C. Desai, Sudhir P. Mudur, and Emil I. Vassev, editors, *Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering (C3S2E'12)*, pages 36–44, New York, NY, USA, June 2010–2013. ACM. ISBN 978-1-4503-1084-0. doi: 10.1145/2347583.2347588. Online e-print `http://arxiv.org/abs/1012.2860`
- Serguei A. Mokhov. *Hybrid Intensional Computing in GIPSY: JLucid, Objective Lucid and GICF*. LAP - Lambert Academic Publishing, March 2010. ISBN 978-3-8383-1198-2
- Serguei A. Mokhov, Joey Paquet, and Mourad Debbabi. Towards automatic deduction and event reconstruction using Forensic Lucid and probabilities to encode the IDS evidence. In S. Jha, R. Sommer, and C. Kreibich, editors, *Proceedings of RAID'10*, LNCS 6307, pages 508–509. Springer, September 2010. doi: 10.1007/978-3-642-15512-3_36
- Serguei A. Mokhov, Joey Paquet, and Mourad Debbabi. The need to support of data flow graph visualization of Forensic Lucid programs, forensic evidence, and their evaluation by GIPSY. [online], September 2010. Poster at VizSec'10; online at `http://arxiv.org/abs/1009.5423`
- Serguei A. Mokhov, Emil Vassev, Joey Paquet, and Mourad Debbabi. Towards a self-forensics property in the ASSL toolset. In *Proceedings of C3S2E'10*, pages 108–113. ACM, May 2010. ISBN 978-1-60558-901-5. doi: 10.1145/1822327.1822342

- 2009

- Joey Paquet. Distributed eductive execution of hybrid intensional programs. In *Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC'09)*, pages 218–224, Seattle, Washington, USA, July 2009. IEEE Computer Society. ISBN 978-0-7695-3726-9
- Serguei A. Mokhov, Joey Paquet, and Xin Tong. A type system for hybrid intensional-imperative programming support in GIPSY. In *Proceedings of C3S2E'09*, pages 101–107, New York, NY, USA, May 2009. ACM. ISBN 978-1-60558-401-0. doi: 10.1145/1557626.1557642

  - Ai Hua Wu. *OO-IP Hybrid Language Design and a Framework Approach to the GIPC*. PhD thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, 2009
  - Serguei A. Mokhov, Joey Paquet, and Mourad Debbabi. Towards automated deduction in blackmail case analysis with Forensic Lucid. In Joseph S. Gauthier, editor, *Proceedings of the Huntsville Simulation Conference (HSC'09)*, pages 326–333. SCS, October 2009. ISBN 978-1-61738-587-2. Online at `http://arxiv.org/abs/0906.0049`
  - Serguei A. Mokhov, Joey Paquet, and Mourad Debbabi. Reasoning about a simulated printer case investigation with Forensic Lucid. In Joseph S. Gauthier, editor, *Proceedings of the Huntsville Simulation Conference (HSC'09)*, page 45. SCS, October 2009. ISBN 978-1-61738-587-2. Abstract, fully online at `http://arxiv.org/abs/0906.5181`
  - Serguei A. Mokhov and Emil Vassev. Self-forensics through case studies of small to medium software systems. In *Proceedings of IMF'09*, pages 128–141. IEEE Computer Society, September 2009. ISBN 978-0-7695-3807-5. doi: 10.1109/IMF.2009.19
  - Serguei A. Mokhov. The role of self-forensics modeling for vehicle crash investigations and event reconstruction simulation. In Joseph S. Gauthier, editor, *Proceedings of the Huntsville Simulation Conference (HSC'09)*, pages 342–349. SCS, October 2009. ISBN 978-1-61738-587-2. Online at `http://arxiv.org/abs/0905.2449`
  - Serguei A. Mokhov. Enhancing the formal cyberforensic approach with observation modeling with credibility factors and mathematical theory of evidence. [online], also in *;login: vol. 34, no. 6, p. 101*, December 2009. Presented at WIPS at USENIX Security'09, `http://www.usenix.org/events/sec09/wips.html`
  - Serguei A. Mokhov. Towards improving validation, verification, crash investigations, and event reconstruction of flight-critical systems with self-forensics. [online], June 2009. A white paper submitted in response to NASA's RFI NNH09ZEA001L, `http://arxiv.org/abs/0906.1845`, mentioned in `http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20100025593_2010028056.pdf`
  - Manuel Peralta, Supratik Mukhopadhyay, and Ramesh Bharadwaj. Automatic synthesis and deployment of intensional kahn process networks. In Dominik Ślęzak, Tai hoon Kim, Stephen S. Yau, Osvaldo Gervasi, and Byeong-Ho Kang, editors, *Grid and Distributed Computing*, volume 63 of *Communications in Computer and Information Science*, pages 73–87. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-10548-7. doi: 10.1007/978-3-642-10549-4_10

- 2008

  - Khaled M. Ben Hamed. *Multidimensional Programs on Distributed Parallel Computers: Analysis and Implementation*. PhD thesis, Computer Science, the University of New Brunswick, February 2008
  - Amir Hossein Pourteymour. Comparative study of Demand Migration Framework implementation using JMS and Jini. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, September 2008
  - Xin Tong. Design and implementation of context calculus in the GIPSY. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, April 2008

- Serguei 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)*, pages 1288–1294, Turku, Finland, July 2008. IEEE Computer Society. doi: 10.1109/COMPSAC.2008.206
- Joey Paquet, Serguei A. Mokhov, and Xin Tong. Design and implementation of context calculus in the GIPSY environment. In *Proceedings of the 32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC)*, pages 1278–1283, Turku, Finland, July 2008. IEEE Computer Society. doi: 10.1109/COMPSAC.2008.200
- John Plaice, Blanca Mancilla, Gabriel Ditu, and William W. Wadge. Sequential demand-driven evaluation of eager TransLucid. In *Proceedings of the 32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC)*, pages 1266–1271, Turku, Finland, July 2008. IEEE Computer Society. doi: 10.1109/COMPSAC.2008.191
- Toby Rahilly and John Plaice. A multithreaded implementation for TransLucid. In *Proceedings of the 32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC)*, pages 1272–1277, Turku, Finland, July 2008. IEEE Computer Society. doi: 10.1109/COMPSAC.2008.191
- Amir Hossein Pourteymour, Emil Vassev, and Joey Paquet. Design and implementation of demand migration systems in GIPSY. In *Proceedings of PDPTA 2009*. CSREA Press, June 2008
- Emil Vassev and Joey Paquet. Towards autonomic GIPSY. In *Proceedings of the Fifth IEEE Workshop on Engineering of Autonomic and Autonomous Systems (EASE 2008)*, pages 25–34. IEEE Computer Society, 2008. ISBN 978-0-7695-3140-3. doi: 10.1109/EASe.2008.9
- Serguei A. Mokhov and Joey Paquet. Formally specifying and proving operational aspects of Forensic Lucid in Isabelle. Technical Report 2008-1-Ait Mohamed, Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada, August 2008. In Theorem Proving in Higher Order Logics (TPHOLs2008): Emerging Trends Proceedings. Online at: `http://users.encs.concordia.ca/~tpholsO8/TPHOLs2008/ET/76-98.pdf` and `http://arxiv.org/abs/0904.3789`
- Serguei A. Mokhov, Joey Paquet, and Mourad Debbabi. Formally specifying operational semantics and language constructs of Forensic Lucid. In Oliver Göbel, Sandra Frings, Detlef Günther, Jens Nedon, and Dirk Schadt, editors, *Proceedings of the IT Incident Management and IT Forensics (IMF'08)*, LNI140, pages 197–216. GI, September 2008. ISBN 978-3-88579-234-5. Online at `http://subs.emis.de/LNI/Proceedings/Proceedings140/gi-proc-140-014.pdf`
- Serguei A. Mokhov. Encoding forensic multimedia evidence from MARF applications as Forensic Lucid expressions. In Tarek Sobh, Khaled Elleithy, and Ausif Mahmood, editors, *Novel Algorithms and Techniques in Telecommunications and Networking, proceedings of CISSE'08*, pages 413–416, University of Bridgeport, CT, USA, December 2008. Springer. ISBN 978-90-481-3661-2. doi: 10.1007/978-90-481-3662-9_71. Printed in January 2010
- Serguei A. Mokhov. Towards security hardening of scientific distributed demand-driven and pipelined computing systems. In *Proceedings of the 7th International*

Symposium on Parallel and Distributed Computing (ISPDC'08), pages 375–382. IEEE Computer Society, July 2008. ISBN 978-0-7695-3472-5. doi: 10.1109/ISPDC.2008.52

- 2007
  - Amir Hossein Pourteymour, Emil Vassev, and Joey Paquet. Towards a new demand-driven message-oriented middleware in GIPSY. In *Proceedings of PDPTA 2007*, pages 91–97. PDPTA, CSREA Press, June 2007
  - Xin Tong, Joey Paquet, and Serguei A. Mokhov. Complete context calculus design and implementation in GIPSY. [online], 2007–2008. `http://arxiv.org/abs/1002.4392`
  - Gabriel Ditu. *The Programming Language TransLucid*. PhD thesis, University of New South Wales, Australia, 2007
- 2006
  - Kaiyu Wan. *Lucx: Lucid Enriched with Context*. PhD thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, 2006
- 2005
  - Kaiyu Wan, Vasu Alagar, and Joey Paquet. Real time reactive programming in lucid enriched with contexts. In Zhiming Liu and Keijiro Araki, editors, *Theoretical Aspects of Computing - ICTAC 2004*, volume 3407 of *Lecture Notes in Computer Science*, pages 387–402. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-25304-4. doi: 10.1007/978-3-540-31862-0_28
  - Kaiyu Wan, Vasu Alagar, and Joey Paquet. Lucx: Lucid enriched with context. In *Proceedings of the 2005 International Conference on Programming Languages and Compilers (PLC 2005)*, pages 48–14. CSREA Press, June 2005
  - Serguei Mokhov and Joey Paquet. General imperative compiler framework within the GIPSY. In *Proceedings of the 2005 International Conference on Programming Languages and Compilers (PLC 2005)*, pages 36–42. CSREA Press, June 2005
  - Serguei A. Mokhov. Towards hybrid intensional programming with JLucid, Objective Lucid, and General Imperative Compiler Framework in the GIPSY. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, October 2005. ISBN 0494102934; online at `http://arxiv.org/abs/0907.2640`
  - Serguei Mokhov and Joey Paquet. Objective Lucid – first step in object-oriented intensional programming in the GIPSY. In *Proceedings of the 2005 International Conference on Programming Languages and Compilers (PLC 2005)*, pages 22–28. CSREA Press, June 2005
  - Peter Grogono, Serguei Mokhov, and Joey Paquet. Towards JLucid, Lucid with embedded Java functions in the GIPSY. In *Proceedings of the 2005 International Conference on Programming Languages and Compilers (PLC 2005)*, pages 15–21. CSREA Press, June 2005

  - Joey Paquet and Ai Hua Wu. GIPSY – a platform for the investigation on intensional programming languages. In *Proceedings of the 2005 International Conference on Programming Languages and Compilers (PLC 2005)*, pages 8–14. CSREA Press, June 2005
  - Emil Vassev and Joey Paquet. A general architecture for demand migration in a demand-driven execution engine in a heterogeneous and distributed environment. In *Proceedings of the 3rd Annual Communication Networks and Services Research Conference (CNSR 2005)*, pages 176–182. IEEE Computer Society, May 2005. doi: 10.1109/CNSR.2005.9
  - Ai Hua Wu and Joey Paquet. Object-oriented intensional programming in the GIPSY: Preliminary investigations. In *Proceedings of the 2005 International Conference on Programming Languages and Compilers (PLC 2005)*, pages 43–47. CSREA Press, June 2005
  - Emil Vassev and Joey Paquet. A generic framework for migrating demands in the GIPSY's demand-driven execution engine. In *Proceedings of the 2005 International Conference on Programming Languages and Compilers (PLC 2005)*, pages 29–35. CSREA Press, June 2005
  - Emil Iordanov Vassev. General architecture for demand migration in the GIPSY demand-driven execution engine. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, June 2005. ISBN 0494102969
  - Kaiyu Wan, Vasu Alagar, and Joey Paquet. A context theory for intensional programming. In *Workshop on Context Representation and Reasoning (CRR05)*, July 2005

- 2004
  - Paul Swoboda. *A Formalisation and Implementation of Distributed Intensional Programming*. PhD thesis, The University of New South Wales, Sydney, Australia, 2004
  - Bo Lu. *Developing the Distributed Component of a Framework for Processing Intensional Programming Languages*. PhD thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, March 2004
  - Paul Swoboda and John Plaice. A new approach to distributed context-aware computing. In A. Ferscha, H. Hoertner, and G. Kotsis, editors, *Advances in Pervasive Computing*. Austrian Computer Society, 2004. ISBN 3-85403-176-9
  - Paul Swoboda and John Plaice. An active functional intensional database. In F. Galindo, editor, *Advances in Pervasive Computing*, pages 56–65. Springer, 2004. LNCS 3180
  - Peter Grogono. Intensional programming in Onyx. Technical report, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, April 2004
  - Joey Paquet, Aihua Wu, and Peter Grogono. Towards a framework for the General Intensional Programming Compiler in the GIPSY. In *Proceedings of the 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2004)*, pages 164–165, New York, NY, USA, October 2004. ACM. doi: 10.1145/1028664.1028731

  - Lei Tao. Warehouse and garbage collection in the GIPSY environment. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, 2004
  - Yimin Ding. Automated translation between graphical and textual representations of intensional programs in the GIPSY. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, June 2004. `http://newton.cs.concordia.ca/~paquet/filetransfer/publications/theses/DingYiminMSc2004.pdf`
  - Nikolaos S. Papaspyrou and Ioannis T. Kassios. GLU# embedded in C++: a marriage between multidimensional and object-oriented programming. *Softw., Pract. Exper.*, 34(7):609–630, 2004. ISSN 0038-0644. doi: 10.1002/spe.582
  - Vasu S. Alagar, Joey Paquet, and Kaiyu Wan. Intensional programming for agent communication. In Jo ao Leite, Andrea Omicini, Paolo Torroni, and Pinar Yolum, editors, *Declarative Agent Languages and Technologies II*, volume 3476 of *Lecture Notes in Computer Science*, pages 239–255. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-26172-8. doi: 10.1007/11493402_14

- 2003
  - Bo Lu, Peter Grogono, and Joey Paquet. Distributed execution of multidimensional programming languages. In *Proceedings of the 15th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2003)*, volume 1, pages 284–289. International Association of Science and Technology for Development, November 2003
  - Ai Hua Wu, Joey Paquet, and Peter Grogono. Design of a compiler framework in the GIPSY system. In *Proceedings of the 15th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2003)*, volume 1, pages 320–328. International Association of Science and Technology for Development, November 2003
  - William W. Wadge. Hammings problem example. [online], December 2003. `http://i.csc.uvic.ca/home/hei/lup/contents.html`
  - Anand Ranganathan and Roy H. Campbell. A middleware for context-aware agents in ubiquitous computing environments. In Markus Endler and Douglas Schmidt, editors, *Proceedings of Middleware 2003*, volume 2672 of *Lecture Notes in Computer Science*, pages 143–161. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-40317-3. doi: 10.1007/3-540-44892-6_8

- 2002
  - Peter Grogono. GIPC increments. Technical report, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, April 2002
  - Ai Hua Wu. Semantic checking and translation in the GIPSY. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, 2002
  - Chun Lei Ren. General intensional programming compiler (GIPC) in the GIPSY. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, 2002

- 2000
  - Joey Paquet and Peter Kropf. The GIPSY architecture. In *Proceedings of Distributed Computing on the Web*, Quebec City, Canada, 2000
- 1999
  - Peter Kropf and John Plaice. Intensional objects. In *International symposium on Languages for Intensional Programming*, pages 37–45, Athens, Greece, June 1999. Demokrits Institute
  - Panagiotis Rondogiannis. Adding multidimensionality to procedural programming languages. *Software: Practice and Experience*, 29(13):1201–1221, 1999. ISSN 1097-024X. doi: 10.1002/(SICI)1097-024X(199911)29:13⟨1201::AID-SPE278⟩3.0.CO;2-0
  - Joey Paquet. *Scientific Intensional Programming*. PhD thesis, Department of Computer Science, Laval University, Sainte-Foy, Canada, 1999
  - Manolis Gergatsoulis and Panos Rondogiannis, editors. *Proceedings of ISLIP'99*, volume Intensional Programming II, June 1999. World Scientific. ISBN: 981-02-4095-3
  - William W. Wadge. Intensional logic in context. In Gergatsoulis and Rondogiannis [33], pages 1–13. Tutorial
  - Thom Frühwirth. Constraint solving with constraint handling rules. In Gergatsoulis and Rondogiannis [33], pages 14–30. Tutorial
  - William W. Wadge and M. C. Schraefel. Putting the hyper back in hypertext. In Gergatsoulis and Rondogiannis [33], pages 31–39
  - M. C. Schraefel, Blanca Mancilla, and John Plaice. Intensional hypertext. In Gergatsoulis and Rondogiannis [33], pages 40–54
  - Yu Zhang and Kang Zhang. Associative query for multi-version web documents. In Gergatsoulis and Rondogiannis [33], pages 55–64
  - Jiannong Cao, Alvin Chan, and Kang Zhang. Programming dynamically reconfigurable web server groups using the DyGOP model. In Gergatsoulis and Rondogiannis [33], pages 65–77
  - Alessandra Raffaetà and Thom Frühwirth. Two semantics for temporal annotated constraint logic. In Gergatsoulis and Rondogiannis [33], pages 78–92
  - Michael Fisher and Tony Kakoudakis. Flexible agent grouping in executable temporal logic. In Gergatsoulis and Rondogiannis [33], pages 93–105
  - Costas D. Koutras and Christos Nomikos. On the computational complexity of stratified negation in linear-time temporal logic programming. In Gergatsoulis and Rondogiannis [33], pages 106–117
  - Manolis Gergatsoulis. Extensions of the branching-time logic programming language *Cactus*. In Gergatsoulis and Rondogiannis [33], pages 118–132
  - Themis Panayiotopoulos. Temporal reasoning with TRL. In Gergatsoulis and Rondogiannis [33], pages 133–148
  - Loïc Besnard, Patricia Bourani, Thierry Gautier, Nicolas Halbwachs, Simin Nadjm-Tehrani, and Annie Ressouche. Design of a multi-formalism application and distribution in a data-flow context: An example. In Gergatsoulis and Rondogiannis [33], pages 149–167

  - Jean-Raymond Gagné and John Plaice. Demand-driven real-time computing. In Gergatsoulis and Rondogiannis [33], pages 168–181. ISBN: 981-02-4095-3
  - Weiqiang Lin and Mehmet A. Orgun. Applied hidden periodicity analysis for mining descrete-valued time series databases. In Gergatsoulis and Rondogiannis [33], pages 182–196
  - Ion Androutsopoulos. Temporal meaning representation in a natural language front-end. In Gergatsoulis and Rondogiannis [33], pages 197–213
  - Win Maung, Chit Swe, and Mehmet A. Orgun. Statistical queries on historical relational databases. In Gergatsoulis and Rondogiannis [33], pages 214–228
  - Themis Panayiyotopoulos and L. C. Baxevanaki. Statistical queries on historical relational databases. In Gergatsoulis and Rondogiannis [33], pages 229–243
  - Weichang Du. Toward an intensional model for programming large scale distributed systems. In Gergatsoulis and Rondogiannis [33], pages 244–258
  - Joey Paquet and John Plaice. The semantics of dimensions as values. In Gergatsoulis and Rondogiannis [33], pages 259–273
  - Panos Rondogiannis. Adding multidimensionality to procedural programming languages. In Gergatsoulis and Rondogiannis [33], pages 274–291
  - John Plaice and Peter G. Kropf. Intensional communities. In Gergatsoulis and Rondogiannis [33], pages 292–295
  - Paul Swoboda and William W. Wadge. Vmake and ISE general tools for the intensionalization of software systems. In Gergatsoulis and Rondogiannis [33], pages 310–320. ISBN: 981-02-4095-3

- 1998

  - William W. Wadge, G. Brown, M. C. Schraefel, and T. Yildirim. Intensional HTML. In *4th International Workshop PODDP'98*, March 1998

- 1997

  - Q. Zhao. Implementation of an object-oriented intensional programming system. Master's thesis, Department of Computer Science, University of New Brunswick, Canada, 1997
  - Raganswamy Jagannathan, Chris Dodd, and Iskender Agi. GLU: A high-level system for granular data-parallel programming. In *Concurrency: Practice and Experience*, volume 1, pages 63–83, 1997

- 1996

  - Chris Dodd. *Intensional Programming I*, chapter Rank analysis in the GLU compiler, pages 76–82. Volume Intensional Programming I of Orgun and Ashcroft [80], May 1996. ISBN: 981-02-2400-1
  - Raganswamy Jagannathan and Chris Dodd. GLU programmer's guide. Technical report, SRI International, Menlo Park, California, 1996

- 1995

- Joey Paquet. Relational databases as multidimensional dataflows. Master's thesis, Departement d'Informatique, Université Laval, Québec, Canada, 1995
- Edward A. Ashcroft, Anthony A. Faustini, Rangaswamy Jagannathan, and William W. Wadge. *Multidimensional Programming*. Oxford University Press, London, February 1995. ISBN: 978-0195075977
- Mehmet A. Orgun and Edward A. Ashcroft, editors. *Proceedings of ISLIP'95*, volume Intensional Programming I, May 1995. World Scientific. ISBN: 981-02-2400-1
- Paul Caspi and Pouzet. A functional extension to Lustre. In Orgun and Ashcroft [80], pages 15–29. Invited Contribution
- Edward A. Ashcroft. Multidimensional program verification: Reasoning about programs that deal with multidimensional objects. In Orgun and Ashcroft [80], pages 30–41. Invited Contribution
- David Abramson and Rok Sosič. Relative debugging using multiple program versions. In Orgun and Ashcroft [80], pages 42–55. Invited Contribution
- William W. Wadge. Possible WOOrlds. In Orgun and Ashcroft [80], pages 56–62. Invited Contribution
- R. Jagannathan. Intensional and extensional graphical models for GLU programming. In Orgun and Ashcroft [80], pages 63–75
- Jiannong Cao, Lichucha Fernando, and Kang Zhang. Programming distributed systems based on graphs. In Orgun and Ashcroft [80], pages 83–95
- Yaowei Liu and John Staples. Building logic constructs into procedural programming languages. In Orgun and Ashcroft [80], pages 96–109
- Lorenzo Verdoscia. ALFA fine grain dataflow machine. In Orgun and Ashcroft [80], pages 110–134
- Iskender Agi. GLU for multidimensional signal processing. In Orgun and Ashcroft [80]. URL `citeseer.ist.psu.edu/agi95glu.html`. ISBN: 981-02-2400-1
- John Plaice. Particle in-cell simulation with Lucid. In Orgun and Ashcroft [80], pages 149–161
- Satoshi Yamane. Real-time object-oriented specification and verification. In Orgun and Ashcroft [80], pages 162–185
- Wanli Ma and Mehmet A. Orgun. Verifying MULTRAN programs with temporal logic. In Orgun and Ashcroft [80], pages 186–206
- William W. Wadge and Alan Yoder. The Possible-World Wide Web. In Orgun and Ashcroft [80], pages 207–213
- Joey Paquet and John Plaice. The intensional relation. In Orgun and Ashcroft [80], pages 214–227
- Panos Rondogiannis and William W. Wadge. Extending the intensionalization algorithm to a broader class of higher-order programs. In Orgun and Ashcroft [80], pages 228–233
- Padmanabhan Krishnan. An asynchronous calculus based on absence of actions. In Orgun and Ashcroft [80], pages 234–248
- Richard Buckland. Choice as a first class citizen. In Orgun and Ashcroft [80], pages 249–259

  - Seiki Akama. A meta-level approach to modal logic programming. In Orgun and Ashcroft [80], pages 260–272
  - Tu Van Le. Fuzzy temporal Prolog. In Orgun and Ashcroft [80], pages 273–280
  - Chuchang Liu and Mehmet A. Orgun. Knowledge-based simulation with Chronolog. In Orgun and Ashcroft [80], pages 281–295

- 1994
  - Panagiotis Rondogiannis. *Higher-Order Functional Languages and Intensional Logic*. PhD thesis, Department of Computer Science, University of Victoria, Victoria, Canada, 1994
  - Joey Paquet and John Plaice. On the design of an indexical query language. In *Proceedings of the Seventh International Symposium on Lucid and Intensional Programming*, pages 28–36, 1994
  - S. Tao. *TLucid and Intensional Attribute Grammars*. PhD thesis, Department of Computer Science, Victoria University, Canada, 1994
  - Weichang Du. Object-oriented implementation of intensional language. In *Proceedings of the 7th International Symposium on Lucid and Intensional Programming*, pages 37–45, Menlo Park, California, USA, September 1994. SRI International

- 1993
  - John Plaice and William W. Wadge. A new approach to version control. *IEEE Transactions on Software*, 19(3):268–276, March 1993
  - John Plaice, Ridha Khedri, and Rene Lalement. From abstract time to real time. In *In Proceedings of the Sixth International Symposium on Lucid and Intensional Programming*, pages 83–93, 1993

- 1991
  - Weichang Du. *Indexical Parallel Programming*. PhD thesis, Department of Computer Science, Victoria University, Canada, 1991
  - B. Freeman-Benson. Lobjcid: Objects in Lucid. In *Proceedings of the 1991 Symposium on Lucid and Intensional Programming*, pages 80–87, Menlo Park, California, USA, April 1991. SRI International

- 1990
  - Weichang Du and William W. Wadge. The eductive implementation of a three-dimensional spreadsheet. *Software Practice and Experience*, 20(11):1097–1114, November 1990. ISSN 0038-0644
  - Weichang Du and William W. Wadge. A 3D spreadsheet based on intensional logic. *IEEE Software*, 7(3):78–89, June 1990. doi: 10.1109/52.55232

- 1989
  - Anthony A. Faustini and E. B. Lewis. *Towards a Real-Time Dataflow Language*. IEEE Computer Society, Los Alamitos, CA, USA, 1989. ISBN 0-8186-0819-6

- 1988

  - J. van Benthem. *A Manual of Intensional Logic*. CSLI Publications, Stanford and The University of Chicago Press, 1988. ISBN 0-937073-29-6

- 1987
  - Sally C. Johnson. A strategy for automatically generating programs in the Lucid programming language (NASA technical memorandum). Technical report, NASA, Scientific and Technical Information Office, 1987. ASIN: B000711R3Q
  - Antony A. Faustini and William W. Wadge. An eductive interpreter for the language Lucid. *SIGPLAN Not.*, 22(7):86–91, 1987. ISSN 0362-1340. doi: 10.1145/960114.29659

- 1985
  - William W. Wadge and Edward A. Ashcroft. *Lucid, the Dataflow Programming Language*. Academic Press, London, 1985

- 1982
  - Anthony A. Faustini. *The Equivalence of a Denotational and an Operational Semantics of Pure Dataflow*. PhD thesis, University of Warwick, Computer Science Department, Coventry, United Kingdom, 1982
  - Edward A. Ashcroft and William W. Wadge. *R* for semantics. *ACM Transactions on Programming Languages and Systems*, 4(2):283–294, April 1982

- 1981
  - C. B. Ostrum. *The Luthid 1.0 Manual*. Department of Computer Science, University of Waterloo, Ontario, Canada, 1981

- 1977
  - Edward A. Ashcroft and William W. Wadge. Lucid, a nonprocedural language with iteration. *Communications of the ACM*, 20(7):519–526, July 1977. ISSN 0001-0782. doi: 10.1145/359636.359715
  - Edward A. Ashcroft and William W. Wadge. Erratum: Lucid – a formal system for writing and proving programs. *SIAM J. Comput.*, 6(1):200, 1977

- 1976
  - Edward A. Ashcroft and William W. Wadge. Lucid – a formal system for writing and proving programs. *SIAM J. Comput.*, 5(3), 1976

Wikipedia and other Wiki entries:

- `http://en.wikipedia.org/wiki/Lucid_(programming_language)`
- `http://en.wikipedia.org/wiki/Category:Intensional_programming_languages`
- `http://en.wikipedia.org/wiki/Intensional_logic`
- `http://www.haskell.org/haskellwiki/Lucid`

3 Core Lucid Standard Specification Design

The Core Lucid standard design and specification is an ongoing process influenced by the two core proposals: GIPL and TransLucid developed in 1999 and 2008 respectively.

3.1 SIGLUCID Meetings

Here's the brief summary of the SIGLUCID meetings at various workshops and conferences, attendees, and works contributing to the collaboration and developing the Core Lucid standard.

3.1.1 SECASA 2010 Meeting at SERA 2010, Montreal, Canada

Works The following works were presented:

1. [39]
2. [69]
3. [145]
4. [68]

TODO

Attendees

1. Serguei A. Mokhov
2. Joey Paquet
3. Emil Vassev
4. Bin Han

TODO

3.1.2 SECASA 2009 Meeting at COMPSAC 2009, Seattle, USA

Works The following works were presented:

1. [87]

TODO

Attendees

1. Joey Paquet
2. John Plaice

TODO

3.1.3 SECASA 2008 Meeting at COMPSAC 2008, Turku, Finland

The first discussion about standardizing the types, evaluation, and overview of the current candidates for the Lucid Core from different research groups, such as GIPL, TransLucid. The needs of various in-progress Lucid dialects were discussed to be accommodated in the core, such as MARFL, Forensic Lucid. Below are the points from the meeting minutes.

Works The following works were presented:

1. [95]
2. [102]
3. [108]
4. [62]

Attendees

1. Weichang Du
2. Blanca Mancilla
3. Serguei A. Mokhov
4. Joey Paquet
5. John Plaice
6. Toby Rahilly
7. William W. Wadge

Notes

1. Constants appearing in the expressions:

```
type<string> (const type -- int8<42> != int16<42>)
[| string |]
12
true
false
```

2. header – default types for integers, etc.
3. #parens – see the sections on the GIPSY type system and a hybrid program example Section 3.3.1 and Section 3.3.2
4. Proposal of `type` ('`type`' is a keyword).
   - type<float 32>
   - uchar<> uchar<>
   - shorthand syntactical sugar:
     `[||| 1.2 |||]` – 64 bit IEEE float
     `{{{ 1.2 }}}` – 32 bit float
5. `special<...>` – correspond to exceptions and error situations for handling later on
6. Joey: Dimensions syntactically are allowed to taken on default values other than always implicit default of zero.
   - `special<undecl>` special¡arith¿ = special¡undecl¿+
     – lose details, e.g. where it happened in the code or even within the imperative code?
   - `if(isspecial<undecl> E)`
     `then ...`
7. Bill has done something like that with someone in pLucid, with well defined semantics, etc.
8. Bill complained about eagerness:

   $$\{E_1 : E_2, ..., E_n : E_{n_2}\}$$

   eager: only lefts (multithreaded [108]), else all —, but right-hand-sides are lazy.
9. Q: How to stop people from producing recursive/infinite contexts?
10. Bill: risky: `\# a@{a:P, E:Q} != P` ??? If E does not terminate or `special` – can't prove it's constant.
11. Toby: threading, sequential scheduling
12. Audience concluded: GIPL context-eager, TransLucid dimension-eager (LHS)
13. Variables

    ```
    variable x
    dimension d
    ```

    type of x is context-dependent.

    Future:

    id<x>

    dimension<d>

    expr<E>

14. Dynamic ranks analysis (Tony Faustini and Bill in the '80)

    $$(X, C)?$$

    ? = eduction evaluation engine

    ```
    W?(X, {})
     ->42
     ->{d1,....,dn}?

    v1=C(d1)
    vn=C{dn}
    v'1=C(d1')
    v'm=C{dm'}

    W?(X, {d1:v1, ....})
     ->42
     -> {d'2,...,d'n}?

    x, W, C, Cs, Ci # 3
    W' = W U {(X, Cs) |-> {3}?}
    ```

    C_i – current internal context

15. Toby: optimization: demand grouping demands as early as possible, lifting up
16. optimization for constant vs. run-time dimensions thus

    `dimension d`

    is an optimization hint.
17. Binary representation (portable) ??? a-la Java byte-code

3.1.4 PLC 2005 Meeting at WORLDCOMP 2005, Las Vegas, USA

Works The following works were presented:

1. [36]
2. [58]
3. [57]
4. [126]
5. [143]
6. [92]
7. [139]

TODO

Attendees

1. Weichang Du
2. Serguei A. Mokhov
3. Joey Paquet
4. Emil Vassev
5. William W. Wadge
6. Kaiyu Wan
7. Aihua Wu

TODO

3.2 TransLucid

TODO

3.3 GIPSY

3.3.1 Hybrid Interaction with Other Languages

GIPC Preprocessor The `Preprocessor` [59, 57] is something that is invoked first by the GIPC (see Figure 1) on incoming GIPSY program's source code stream. The `Preprocessor`'s role is to do preliminary program analysis, processing, and splitting the source GIPSY program into "chunks", each written in a different language and identified by a *language tag*. In a very general view, a GIPSY program is a hybrid program consisting of different languages in one or more source file; then, there has to be an interface between all these code segments. Thus, the `Preprocessor` after some initial parsing (using its own preprocessor syntax) and producing the initial parse tree, constructs a preliminary dictionary of symbols used throughout the program. This is the basis for type matching and semantic analysis applied later on. This is also where the first step of type assignment occurs, especially on the boundary between typed and typeless parts of the program, e.g. Java and a specific Lucid dialect. The `Preprocessor` then splits the code segments of the GIPSY program into chunks preparing them to be fed to the respective concrete compilers for those chunks. The chunks are represented through the `CodeSegment` class that the `GIPC` collects.

GIPSY Program Segments There are four baseline types of segments defined to be used in a GIPSY program. These are:

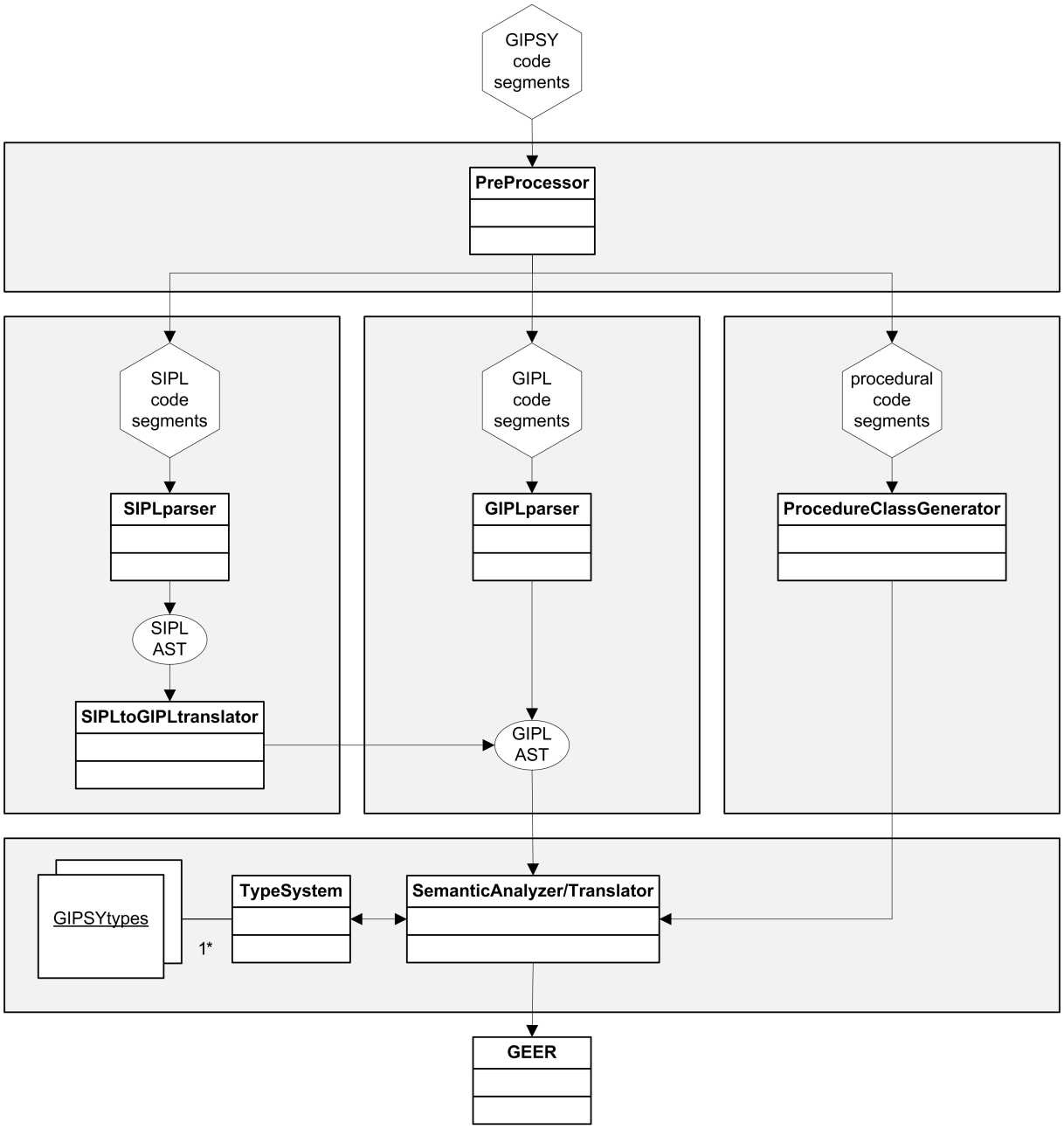

Figure 1: GIPC Framework with the Preprocessor

- `#funcdecl` program segment declares function prototypes written as imperative language functions defined later or externally from this program to be used by the intensional language part. The syntactical form of these prototypes is particular to GIPSY programs and need not resemble the actual function prototype declaration they describe in their particular programming language. They serve as a basis for static and dynamic type assignment and checking within the GIPSY type system with regards to procedural functions called by other parts of the GIPSY program, e.g. the Lucid code segments.

- `#typedecl` segment lists all user-defined data types that can potentially be used by the

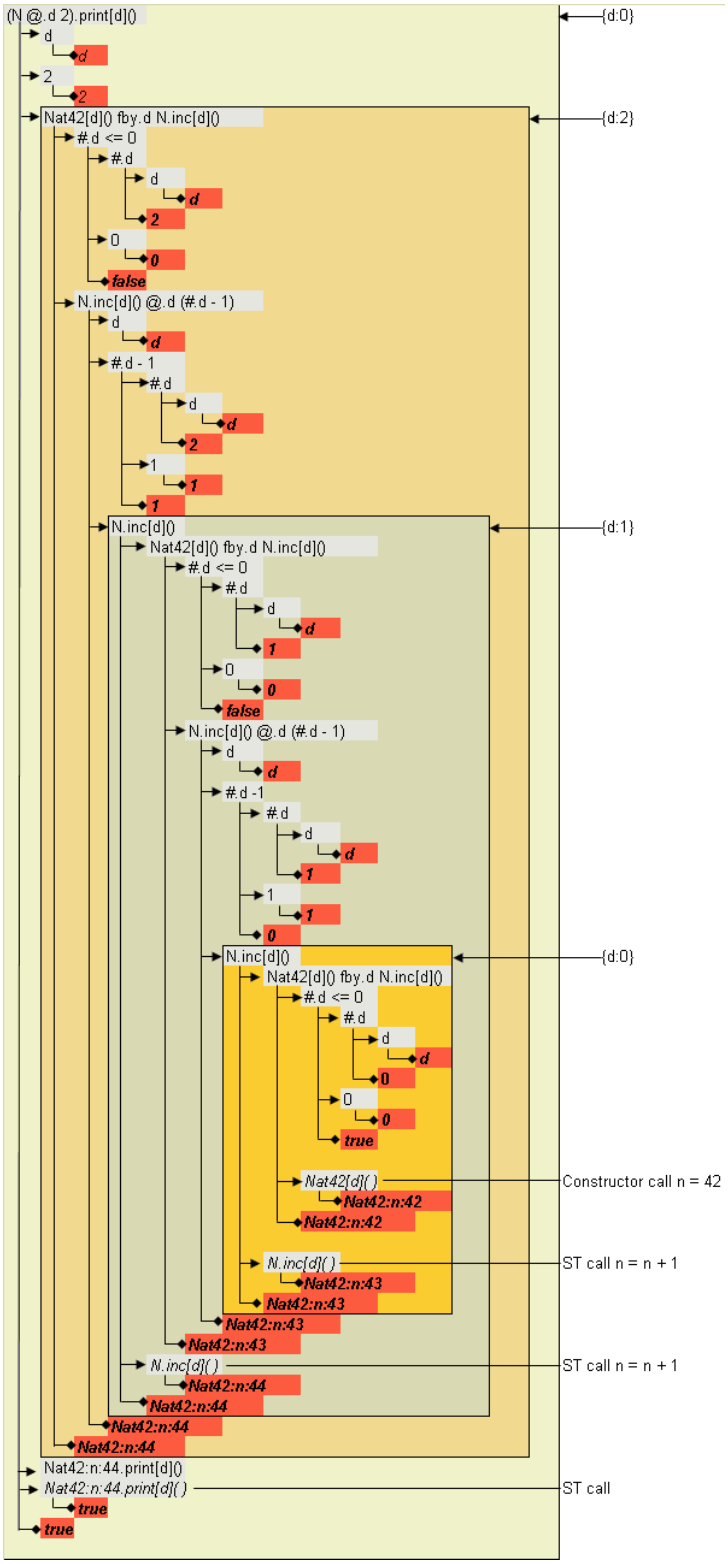

Figure 2: Example of Eductive Evaluation of Objective Lucid Progran

intensional part; usually objects. These are the types that do not explicitly appear in the matching table in Table 1 describing the basic data types allowed in GIPSY programs.

- `#<IMPERATIVELANG>` segment declares that this is a code segment written in whatever IMPERATIVELANG may be, for example `#JAVA` for Java, `#CPP` for C++, `#FORTRAN` for Fortran, `#PERL` for Perl, `#PYTHON` for Python, etc.

- `#<INTENSIONALLANG>` segment declares that this is a code segment written in whatever INTENSIONALLANG may be, for example `#GIPL`, `#LUCX`, `#JOOIP`, `#INDEXICALLUCID`, `#JLUCID`, `#OBJECTIVELUCID`, `#TENSORLUCID`, `#ONYX` [35], `#FORENSICLUCID` [67], and `#TRANSLUCID`, etc. as specified by the available GIPSY implementations and stubs. An example of a hybrid program is presented in Listing 1. The preamble of the program with the type and function declaration segments are the main source of type information that is used at compile time to annotate the nodes in the tree to help both static and semantic analysis.

3.3.2 Introduction to the GIPSY Type System

The introduction of JLucid, Objective Lucid, and GICF [59, 58, 57, 36] prompted the development of the GIPSY Type System as implicitly understood by the Lucid language and its incarnation within the GIPSY to handle types in a more general manner as a glue between the imperative and intensional languages within the system. Further evolution of Lucx introducing contexts as first-class values and JOOIP highlighted the need of the further development of the type system to accommodate the more general properties of the intensional and hybrid languages.

Matching Lucid and Java Data Types Here we present a case of interaction between Lucid and Java. Allowing Lucid to call Java methods brings a set of issues related to the data types, especially when it comes to type checks between Lucid and Java parts of a hybrid program. This is pertinent when Lucid variables or expressions are used as parameters to Java methods and when a Java method returns a result to be assigned to a Lucid variable or used in an intensional expression. The sets of types in both cases are not exactly the same. The basic set of Lucid data types as defined by Grogono [34] is `int`, `bool`, `double`, `string`, and `dimension`. Lucid's `int` is of the same size as Java's `long`. GIPSY and Java `double`, `boolean`, and `String` are roughly the same. Lucid `string` and Java `String` are simply mapped internally through `StringBuffer`; thus, one can think of the Lucid `string` as a reference when evaluated in the intensional program. Based on this fact, the lengths of a Lucid `string` and Java `String` are the same. Java `String` is also an object in Java; however, at this point, a Lucid program has no direct access to any `String`'s properties (though internally we do and we may expose it later to the programmers). We also distinguish the `float` data type for single-precision floating point operations. The `dimension` index type is said to be an integer or string (as far as its dimension tag values are concerned), but might be of other types eventually, as discussed in [123]. Therefore, we perform data type matching as presented in Table 1. Additionally, we allow `void` Java return type which will always be matched to a Boolean expression `true` in Lucid as an expression has to always evaluate to something. As for now our types mapping and restrictions are as per Table 1. This is the mapping table for the Java-to-IPL-to-Java type adapter. Such a table would exist for mapping between any imperative-to-intensional language and back, e.g. the C++-to-IPL-to-C++ type adapter.

```
/**
 * Language-mix GIPSY program.
 * @author Serguei Mokhov
 */
#typedecl

myclass;

#funcdecl

myclass foo(int,double);
float bar(int,int):"ftp://newton.cs.concordia.ca/cool.class":baz;
int f1();

#JAVA
myclass foo(int a, double b)
{
   return new myclass(new Integer((int)(b + a)));
}

class myclass
{
   public myclass(Integer a)
   {
      System.out.println(a);
   }
}

#CPP
#include <iostream>

int f1(void)
{
   cout << "hello";
   return 0;
}

#OBJECTIVELUCID

A + bar(B, C)
where
   A = foo(B, C).intValue();
   B = f1();
   C = 2.0;
end;

/*
 * in theory we could write more than one intensional chunk,
 * then those chunks would evaluate as separate possibly
 * totally independent expressions in parallel that happened
 * to use the same set of imperative functions.
 */

// EOF
```

Listing 1: Example of a hybrid GIPSY program.

Table 1: Matching data types between Lucid and Java.

Return Types of Java Methods	Types of Lucid Expressions	Internal GIPSY Types
`int`, `byte`, `long`	`int`, `dimension`	`GIPSYInteger`
`float`	`float`	`GIPSYFloat`
`double`	`double`	`GIPSYDouble`
`boolean`	`bool`	`GIPSYBoolean`
`char`	`char`	`GIPSYCharacter`
`String`	`string`, `dimension`	`GIPSYString`
`Method`	*function*	`GIPSYFunction`
`Method`	*operator*	`GIPSYOperator`
`[]`	`[]`	`GIPSYArray`
`Object`	*class*	`GIPSYObject`
`Object`	*URL*	`GIPSYEmbed`
`void`	`bool::true`	`GIPSYVoid`
Parameter Types Used in Lucid	**Corresponding Java Types**	**Internal GIPSY Types**
`string`	`String`	`GIPSYString`
`float`	`float`	`GIPSYFloat`
`double`	`double`	`GIPSYDouble`
`int`	`int`	`GIPSYInteger`
`dimension`	`int`, `String`	`Dimension`
`bool`	`boolean`	`GIPSYBoolean`
class	`Object`	`GIPSYObject`
URL	`Object`	`GIPSYEmbed`
`[]`	`[]`	`GIPSYArray`
operator	`Method`	`GIPSYOperator`
function	`Method`	`GIPSYFunction`

Overview of the Design and Implementation of the Type System. While the main language of GIPSY, Lucid, is polymorphic and does not have explicit types, co-existing with other languages necessitates definition of GIPSY types and their mapping to a particular language being embedded. Figure 3 presents the detailed design of the GIPSY Type System.

Each class is prefixed with `GIPSY` to avoid possible confusion with similar definitions in the `java.lang` package. The `GIPSYVoid` type always evaluates to the Boolean `true`, as described earlier in Section 3.3.2. The other types wrap around the corresponding Java object wrapper classes for the primitive types, such as `Long`, `Float`, etc. Every class keeps a lexeme (a lexical representation) of the corresponding type in a GIPSY program and overrides `toString()` to show the lexeme and the contained value. These types are extensively used by the `Preprocessor`, imperative and intensional (for constants) compilers, the `SequentialThreadGenerator`, and `SemanticAnalyzer` for the general type of GIPSY program processing, and by the GEE's `Executor`.

The other special types that have been created are either experimental or do not correspond to a wrapper of a primitive type. `GIPSYIdentifier` type case corresponds to a declaration of some sort of an identifier in a GIPSY program to be put into the dictionary, be it a variable or a function name with the reference to their definition. Constants and conditionals may be anonymous and thereby not have a corresponding identifier. `GIPSYEmbed` is another special type that encapsulates embedded code via the URL parameter and later is exploded into multiple types

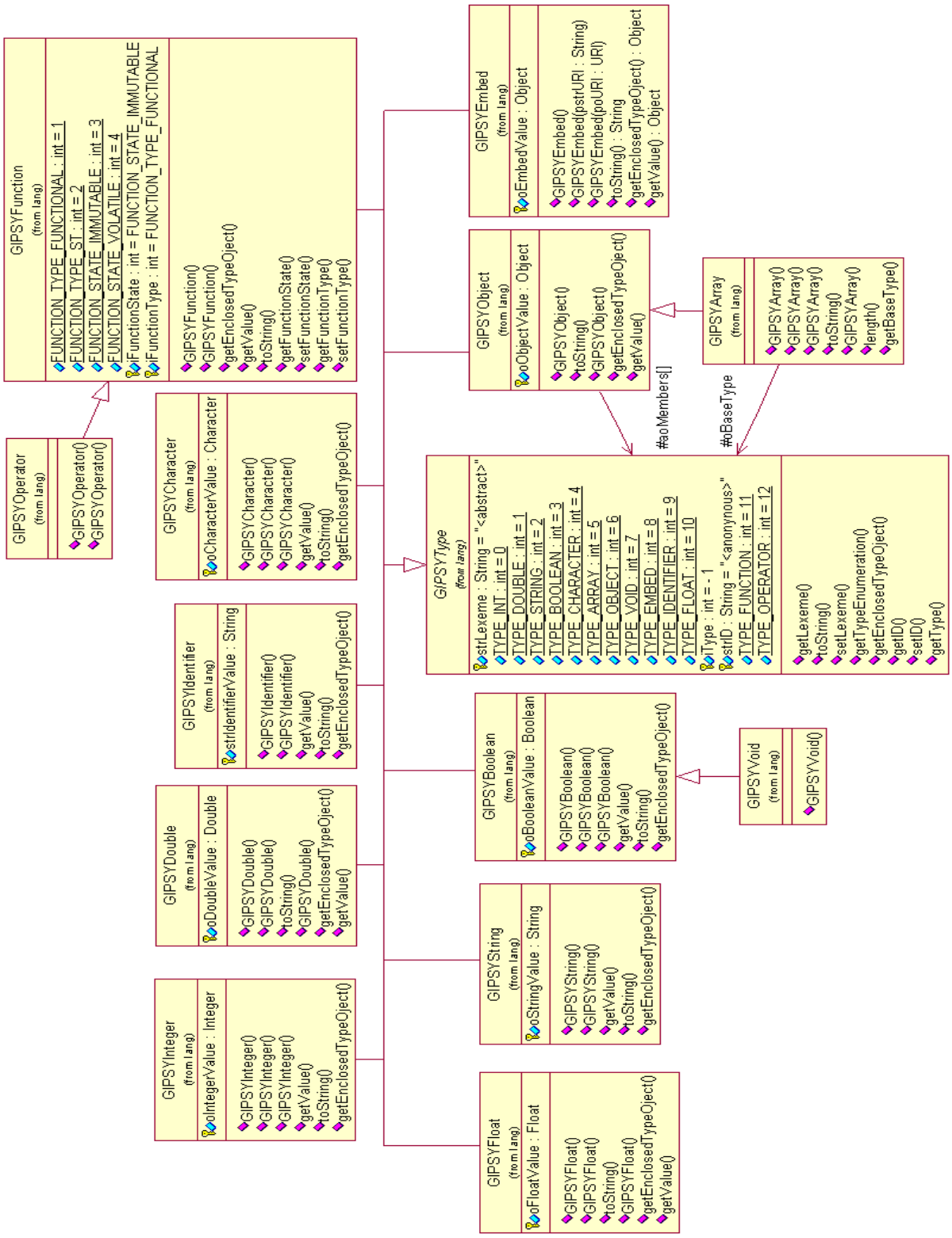

Figure 3: GIPSY Type System.

corresponding to procedural demands (Java or any other language methods or functions) [59, 36]. `GIPSYFunction` and its descendant `GIPSYOperator` correspond to the function types for regular operators and user-defined functions. A `GIPSYFunction` can either encapsulate an ordinary Lucid function (which is immutable as in functional programming) or a procedure (e.g. a Java method), which may often be mutable (i.e. with side effects). These four types (*identifier, embed, function*, and *operator*) are not directly exposed to a GIPSY programmer and at this point are managed internally. By the latter we mean we have not reached the stage when we can provide them for explicit use by programmers; however, the semantics of is still defined and specified at the requirements, design, and implementation levels. `GIPSYContext` and `Dimension` are a new addition to the type system implementation since [59]. They represent context-as-first-class-values in the context calculus defined by Wan in [137] and refined and implemented by Tong [122]. The rest of the type system is exposed to the GIPSY programmer in the preamble of a GIPSY program, i.e., the `#funcdecl` and `#typedecl` segments, which result in the embryo of the dictionary for linking, semantic analysis, and execution. Once imperative compilers of procedural demands return, the type data structures (return and parameter types) declared in the preamble are matched against what was discovered by the compilers and if the match is successful, the link is made. By capturing the types such as *identifier, embed, function, operator* and *context, dimension*, the GIPSY type system lays down fundamentals the higher-order intensional logic (HOIL) support that combines functional programming, intensional logic, context calculus, and in some instances hybrid paradigm support, and the corresponding types. We describe various properties of the concrete GIPSY types and their more detailed specification in Appendix **??** and Appendix **??**. There we detail the inner workings of each type in more detail as well describe some of the properties through the notions of existential, union, intersection, and linear types. They have been excluded from the main body of the article due to size limitations.

3.4 The Core Lucid Standard

The Core Lucid standard specification, syntax, semantics, translation rules, type system, and verifications are to be placed in this section upon consensus of the SIGLUCID members.

TODO

3.4.1 Syntax

TODO

3.4.2 Semantics

TODO

4 Conclusion

We have layed out the first foundational notions of a practical Lucid standard at the 1st SECASA in 2008 in Turku, Finland, associated with COMPSAC 2008. Since then two (3) more SECASA's happened: in 2009 in Seattle, 2010 in Montreal, and another one is planned in 2011. Prior that we are producing this first set of notes from the meeting and related work.

5 Future Work

We plan on further meet and refine these notes and the standards and further accrete the related work. Our eventual goal after the standard draft is complete publish it along with a comprehensive survey of the recent related work as well as historical review.

6 Acknowledgments

This research work was funded by the respective grants, faculties and departments of the authors. Acknowledgment also goes to the former and present colleagues collaborated with us in the past.

References

[1] David Abramson and Rok Sosič. Relative debugging using multiple program versions. In Orgun and Ashcroft [80], pages 42–55. Invited Contribution.

[2] Iskender Agi. GLU for multidimensional signal processing. In Orgun and Ashcroft [80]. URL `citeseer.ist.psu.edu/agi95glu.html`. ISBN: 981-02-2400-1.

[3] Seiki Akama. A meta-level approach to modal logic programming. In Orgun and Ashcroft [80], pages 260–272.

[4] Vasu S. Alagar, Joey Paquet, and Kaiyu Wan. Intensional programming for agent communication. In Jo ao Leite, Andrea Omicini, Paolo Torroni, and Pinar Yolum, editors, *Declarative Agent Languages and Technologies II*, volume 3476 of *Lecture Notes in Computer Science*, pages 239–255. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-26172-8. doi: 10.1007/11493402_14.

[5] Ion Androutsopoulos. Temporal meaning representation in a natural language front-end. In Gergatsoulis and Rondogiannis [33], pages 197–213.

[6] Edward A. Ashcroft. Multidimensional program verification: Reasoning about programs that deal with multidimensional objects. In Orgun and Ashcroft [80], pages 30–41. Invited Contribution.

[7] Edward A. Ashcroft and William W. Wadge. Lucid – a formal system for writing and proving programs. *SIAM J. Comput.*, 5(3), 1976.

[8] Edward A. Ashcroft and William W. Wadge. Erratum: Lucid – a formal system for writing and proving programs. *SIAM J. Comput.*, 6(1):200, 1977.

[9] Edward A. Ashcroft and William W. Wadge. Lucid, a nonprocedural language with iteration. *Communications of the ACM*, 20(7):519–526, July 1977. ISSN 0001-0782. doi: 10.1145/359636.359715.

[10] Edward A. Ashcroft and William W. Wadge. *R* for semantics. *ACM Transactions on Programming Languages and Systems*, 4(2):283–294, April 1982.

[11] Edward A. Ashcroft, Anthony A. Faustini, Rangaswamy Jagannathan, and William W. Wadge. *Multidimensional Programming*. Oxford University Press, London, February 1995. ISBN: 978-0195075977.

[12] Loïc Besnard, Patricia Bourani, Thierry Gautier, Nicolas Halbwachs, Simin Nadjm-Tehrani, and Annie Ressouche. Design of a multi-formalism application and distribution in a data-flow context: An example. In Gergatsoulis and Rondogiannis [33], pages 149–167.

[13] Richard Buckland. Choice as a first class citizen. In Orgun and Ashcroft [80], pages 249–259.

[14] Jiannong Cao, Lichucha Fernando, and Kang Zhang. Programming distributed systems based on graphs. In Orgun and Ashcroft [80], pages 83–95.

[15] Jiannong Cao, Alvin Chan, and Kang Zhang. Programming dynamically reconfigurable web server groups using the DyGOP model. In Gergatsoulis and Rondogiannis [33], pages 65–77.

[16] Paul Caspi and Pouzet. A functional extension to Lustre. In Orgun and Ashcroft [80], pages 15–29. Invited Contribution.

[17] Yimin Ding. Automated translation between graphical and textual representations of intensional programs in the GIPSY. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, June 2004. `http://newton.cs.concordia.ca/~paquet/filetransfer/publications/theses/DingYiminMSc2004.pdf`.

[18] Gabriel Ditu. *The Programming Language TransLucid*. PhD thesis, University of New South Wales, Australia, 2007.

[19] Chris Dodd. *Intensional Programming I*, chapter Rank analysis in the GLU compiler, pages 76–82. Volume Intensional Programming I of Orgun and Ashcroft [80], May 1996. ISBN: 981-02-2400-1.

[20] Weichang Du. *Indexical Parallel Programming*. PhD thesis, Department of Computer Science, Victoria University, Canada, 1991.

[21] Weichang Du. Object-oriented implementation of intensional language. In *Proceedings of the 7th International Symposium on Lucid and Intensional Programming*, pages 37–45, Menlo Park, California, USA, September 1994. SRI International.

[22] Weichang Du. Toward an intensional model for programming large scale distributed systems. In Gergatsoulis and Rondogiannis [33], pages 244–258.

[23] Weichang Du and William W. Wadge. A 3D spreadsheet based on intensional logic. *IEEE Software*, 7(3):78–89, June 1990. doi: 10.1109/52.55232.

[24] Weichang Du and William W. Wadge. The eductive implementation of a three-dimensional spreadsheet. *Software Practice and Experience*, 20(11):1097–1114, November 1990. ISSN 0038-0644.

[25] Anthony A. Faustini. *The Equivalence of a Denotational and an Operational Semantics of Pure Dataflow*. PhD thesis, University of Warwick, Computer Science Department, Coventry, United Kingdom, 1982.

[26] Anthony A. Faustini and E. B. Lewis. *Towards a Real-Time Dataflow Language*. IEEE Computer Society, Los Alamitos, CA, USA, 1989. ISBN 0-8186-0819-6.

[27] Antony A. Faustini and William W. Wadge. An eductive interpreter for the language Lucid. *SIGPLAN Not.*, 22(7):86–91, 1987. ISSN 0362-1340. doi: 10.1145/960114.29659.

[28] Michael Fisher and Tony Kakoudakis. Flexible agent grouping in executable temporal logic. In Gergatsoulis and Rondogiannis [33], pages 93–105.

[29] B. Freeman-Benson. Lobjcid: Objects in Lucid. In *Proceedings of the 1991 Symposium on Lucid and Intensional Programming*, pages 80–87, Menlo Park, California, USA, April 1991. SRI International.

[30] Thom Frühwirth. Constraint solving with constraint handling rules. In Gergatsoulis and Rondogiannis [33], pages 14–30. Tutorial.

[31] Jean-Raymond Gagné and John Plaice. Demand-driven real-time computing. In Gergatsoulis and Rondogiannis [33], pages 168–181. ISBN: 981-02-4095-3.

[32] Manolis Gergatsoulis. Extensions of the branching-time logic programming language *Cactus*. In Gergatsoulis and Rondogiannis [33], pages 118–132.

[33] Manolis Gergatsoulis and Panos Rondogiannis, editors. *Proceedings of ISLIP'99*, volume Intensional Programming II, June 1999. World Scientific. ISBN: 981-02-4095-3.

[34] Peter Grogono. GIPC increments. Technical report, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, April 2002.

[35] Peter Grogono. Intensional programming in Onyx. Technical report, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, April 2004.

[36] Peter Grogono, Serguei Mokhov, and Joey Paquet. Towards JLucid, Lucid with embedded Java functions in the GIPSY. In *Proceedings of the 2005 International Conference on Programming Languages and Compilers (PLC 2005)*, pages 15–21. CSREA Press, June 2005.

[37] Khaled M. Ben Hamed. *Multidimensional Programs on Distributed Parallel Computers: Analysis and Implementation*. PhD thesis, Computer Science, the University of New Brunswick, February 2008.

[38] Bin Han. Towards a multi-tier runtime system for GIPSY. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, 2010.

[39] Bin Han, Serguei A. Mokhov, and Joey Paquet. Advances in the design and implementation of a multi-tier architecture in the GIPSY environment with Java. In *Proceedings of SERA 2010*, pages 259–266. IEEE Computer Society, 2010. ISBN 978-0-7695-4075-7. doi: 10.1109/SERA.2010.40. Online at `http://arxiv.org/abs/0906.4837`.

[40] R. Jagannathan. Intensional and extensional graphical models for GLU programming. In Orgun and Ashcroft [80], pages 63–75.

[41] Raganswamy Jagannathan and Chris Dodd. GLU programmer's guide. Technical report, SRI International, Menlo Park, California, 1996.

[42] Raganswamy Jagannathan, Chris Dodd, and Iskender Agi. GLU: A high-level system for granular data-parallel programming. In *Concurrency: Practice and Experience*, volume 1, pages 63–83, 1997.

[43] Yi Ji. Scalability evaluation of the GIPSY runtime system. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, March 2011.

[44] Yi Ji, Serguei A. Mokhov, and Joey Paquet. Unifying and refactoring DMF to support concurrent Jini and JMS DMS in GIPSY. In Bipin C. Desai, Sudhir P. Mudur, and Emil I. Vassev, editors, *Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering (C3S2E'12)*, pages 36–44, New York, NY, USA, June 2010–2013. ACM. ISBN 978-1-4503-1084-0. doi: 10.1145/2347583.2347588. Online e-print `http://arxiv.org/abs/1012.2860`.

[45] Sally C. Johnson. A strategy for automatically generating programs in the Lucid programming language (NASA technical memorandum). Technical report, NASA, Scientific and Technical Information Office, 1987. ASIN: B000711R3Q.

[46] Costas D. Koutras and Christos Nomikos. On the computational complexity of stratified negation in linear-time temporal logic programming. In Gergatsoulis and Rondogiannis [33], pages 106–117.

[47] Padmanabhan Krishnan. An asynchronous calculus based on absence of actions. In Orgun and Ashcroft [80], pages 234–248.

[48] Peter Kropf and John Plaice. Intensional objects. In *International symposium on Languages for Intensional Programming*, pages 37–45, Athens, Greece, June 1999. Demokrits Institute.

[49] Tu Van Le. Fuzzy temporal Prolog. In Orgun and Ashcroft [80], pages 273–280.

[50] Weiqiang Lin and Mehmet A. Orgun. Applied hidden periodicity analysis for mining descrete-valued time series databases. In Gergatsoulis and Rondogiannis [33], pages 182–196.

[51] Chuchang Liu and Mehmet A. Orgun. Knowledge-based simulation with Chronolog. In Orgun and Ashcroft [80], pages 281–295.

[52] Yaowei Liu and John Staples. Building logic constructs into procedural programming languages. In Orgun and Ashcroft [80], pages 96–109.

[53] Bo Lu. *Developing the Distributed Component of a Framework for Processing Intensional Programming Languages*. PhD thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, March 2004.

[54] Bo Lu, Peter Grogono, and Joey Paquet. Distributed execution of multidimensional programming languages. In *Proceedings of the 15th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2003)*, volume 1, pages 284–289. International Association of Science and Technology for Development, November 2003.

[55] Wanli Ma and Mehmet A. Orgun. Verifying MULTRAN programs with temporal logic. In Orgun and Ashcroft [80], pages 186–206.

[56] Win Maung, Chit Swe, and Mehmet A. Orgun. Statistical queries on historical relational databases. In Gergatsoulis and Rondogiannis [33], pages 214–228.

[57] Serguei Mokhov and Joey Paquet. General imperative compiler framework within the GIPSY. In *Proceedings of the 2005 International Conference on Programming Languages and Compilers (PLC 2005)*, pages 36–42. CSREA Press, June 2005.

[58] Serguei Mokhov and Joey Paquet. Objective Lucid – first step in object-oriented intensional programming in the GIPSY. In *Proceedings of the 2005 International Conference on Programming Languages and Compilers (PLC 2005)*, pages 22–28. CSREA Press, June 2005.

[59] Serguei A. Mokhov. Towards hybrid intensional programming with JLucid, Objective Lucid, and General Imperative Compiler Framework in the GIPSY. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, October 2005. ISBN 0494102934; online at `http://arxiv.org/abs/0907.2640`.

[60] Serguei A. Mokhov. Towards security hardening of scientific distributed demand-driven and pipelined computing systems. In *Proceedings of the 7th International Symposium on Parallel and Distributed Computing (ISPDC'08)*, pages 375–382. IEEE Computer Society, July 2008. ISBN 978-0-7695-3472-5. doi: 10.1109/ISPDC.2008.52.

[61] Serguei A. Mokhov. Encoding forensic multimedia evidence from MARF applications as Forensic Lucid expressions. In Tarek Sobh, Khaled Elleithy, and Ausif Mahmood, editors, *Novel Algorithms and Techniques in Telecommunications and Networking, proceedings of CISSE'08*, pages 413–416, University of Bridgeport, CT, USA, December 2008. Springer. ISBN 978-90-481-3661-2. doi: 10.1007/978-90-481-3662-9_71. Printed in January 2010.

[62] Serguei 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)*, pages 1288–1294, Turku, Finland, July 2008. IEEE Computer Society. doi: 10.1109/COMPSAC.2008.206.

[63] Serguei A. Mokhov. Enhancing the formal cyberforensic approach with observation modeling with credibility factors and mathematical theory of evidence. [online], also in *;login: vol. 34, no. 6, p. 101*, December 2009. Presented at WIPS at USENIX Security'09, `http://www.usenix.org/events/sec09/wips.html`.

[64] Serguei A. Mokhov. The role of self-forensics modeling for vehicle crash investigations and event reconstruction simulation. In Joseph S. Gauthier, editor, *Proceedings of the*

Huntsville Simulation Conference (HSC'09), pages 342–349. SCS, October 2009. ISBN 978-1-61738-587-2. Online at `http://arxiv.org/abs/0905.2449`.

[65] Serguei A. Mokhov. Towards improving validation, verification, crash investigations, and event reconstruction of flight-critical systems with self-forensics. [online], June 2009. A white paper submitted in response to NASA's RFI NNH09ZEA001L, `http://arxiv.org/abs/0906.1845`, mentioned in `http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20100025593_2010028056.pdf`.

[66] Serguei A. Mokhov. *Hybrid Intensional Computing in GIPSY: JLucid, Objective Lucid and GICF*. LAP - Lambert Academic Publishing, March 2010. ISBN 978-3-8383-1198-2.

[67] Serguei A. Mokhov and Joey Paquet. Formally specifying and proving operational aspects of Forensic Lucid in Isabelle. Technical Report 2008-1-Ait Mohamed, Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada, August 2008. In Theorem Proving in Higher Order Logics (TPHOLs2008): Emerging Trends Proceedings. Online at: `http://users.encs.concordia.ca/~tphols08/TPHOLs2008/ET/76-98.pdf` and `http://arxiv.org/abs/0904.3789`.

[68] Serguei A. Mokhov and Joey Paquet. Using the General Intensional Programming System (GIPSY) for evaluation of higher-order intensional logic (HOIL) expressions. In *Proceedings of SERA 2010*, pages 101–109. IEEE Computer Society, May 2010. ISBN 978-0-7695-4075-7. doi: 10.1109/SERA.2010.23. Online at `http://arxiv.org/abs/0906.3911`.

[69] Serguei A. Mokhov and Joey Paquet. A type system for higher-order intensional logic support for variable bindings in hybrid intensional-imperative programs in GIPSY. In Tokuro Matsuo, Naohiro Ishii, and Roger Lee, editors, *9th IEEE/ACIS International Conference on Computer and Information Science, IEEE/ACIS ICIS 2010*, pages 921–928. IEEE Computer Society, May 2010. ISBN 978-0-7695-4147-1. doi: 10.1109/ICIS.2010.156. Presented at SERA 2010; online at `http://arxiv.org/abs/0906.3919`.

[70] Serguei A. Mokhov and Emil Vassev. Self-forensics through case studies of small to medium software systems. In *Proceedings of IMF'09*, pages 128–141. IEEE Computer Society, September 2009. ISBN 978-0-7695-3807-5. doi: 10.1109/IMF.2009.19.

[71] Serguei A. Mokhov, Joey Paquet, and Mourad Debbabi. Formally specifying operational semantics and language constructs of Forensic Lucid. In Oliver Göbel, Sandra Frings, Detlef Günther, Jens Nedon, and Dirk Schadt, editors, *Proceedings of the IT Incident Management and IT Forensics (IMF'08)*, LNI140, pages 197–216. GI, September 2008. ISBN 978-3-88579-234-5. Online at `http://subs.emis.de/LNI/Proceedings/Proceedings140/gi-proc-140-014.pdf`.

[72] Serguei A. Mokhov, Joey Paquet, and Mourad Debbabi. Towards automated deduction in blackmail case analysis with Forensic Lucid. In Joseph S. Gauthier, editor, *Proceedings of the Huntsville Simulation Conference (HSC'09)*, pages 326–333. SCS, October 2009. ISBN 978-1-61738-587-2. Online at `http://arxiv.org/abs/0906.0049`.

[73] Serguei A. Mokhov, Joey Paquet, and Mourad Debbabi. Reasoning about a simulated printer case investigation with Forensic Lucid. In Joseph S. Gauthier, editor, *Proceedings of the Huntsville Simulation Conference (HSC'09)*, page 45. SCS, October 2009. ISBN 978-1-61738-587-2. Abstract, fully online at `http://arxiv.org/abs/0906.5181`.

[74] Serguei A. Mokhov, Joey Paquet, and Xin Tong. A type system for hybrid intensional-imperative programming support in GIPSY. In *Proceedings of C3S2E'09*, pages 101–107, New York, NY, USA, May 2009. ACM. ISBN 978-1-60558-401-0. doi: 10.1145/1557626.1557642.

[75] Serguei A. Mokhov, Joey Paquet, and Mourad Debbabi. The need to support of data flow graph visualization of Forensic Lucid programs, forensic evidence, and their evaluation by GIPSY. [online], September 2010. Poster at VizSec'10; online at `http://arxiv.org/abs/1009.5423`.

[76] Serguei A. Mokhov, Joey Paquet, and Mourad Debbabi. Towards automatic deduction and event reconstruction using Forensic Lucid and probabilities to encode the IDS evidence. In S. Jha, R. Sommer, and C. Kreibich, editors, *Proceedings of RAID'10*, LNCS 6307, pages 508–509. Springer, September 2010. doi: 10.1007/978-3-642-15512-3_36.

[77] Serguei A. Mokhov, Emil Vassev, Joey Paquet, and Mourad Debbabi. Towards a self-forensics property in the ASSL toolset. In *Proceedings of C3S2E'10*, pages 108–113. ACM, May 2010. ISBN 978-1-60558-901-5. doi: 10.1145/1822327.1822342.

[78] Serguei A. Mokhov, Joey Paquet, and Mourad Debbabi. On the need for data flow graph visualization of Forensic Lucid programs and forensic evidence, and their evaluation by GIPSY. In *Proceedings of the Ninth Annual International Conference on Privacy, Security and Trust (PST), 2011*, pages 120–123. IEEE Computer Society, July 2011. ISBN 978-1-4577-0582-3. doi: 10.1109/PST.2011.5971973. Short paper; full version online at `http://arxiv.org/abs/1009.5423`.

[79] Serguei A. Mokhov, Joey Paquet, and Mourad Debbabi. Reasoning about a simulated printer case investigation with Forensic Lucid. In Pavel Gladyshev and Marcus K. Rogers, editors, *Proceedings of ICDF2C'11*, number 0088 in LNICST, pages 282–296. Springer, October 2011. ISBN 978-3-642-35514-1. doi: 10.1007/978-3-642-35515-8_23. Submitted in 2011, appeared in 2012; online at `http://arxiv.org/abs/0906.5181`.

[80] Mehmet A. Orgun and Edward A. Ashcroft, editors. *Proceedings of ISLIP'95*, volume Intensional Programming I, May 1995. World Scientific. ISBN: 981-02-2400-1.

[81] C. B. Ostrum. *The Luthid 1.0 Manual*. Department of Computer Science, University of Waterloo, Ontario, Canada, 1981.

[82] Themis Panayiotopoulos. Temporal reasoning with TRL. In Gergatsoulis and Rondogiannis [33], pages 133–148.

[83] Themis Panayiyotopoulos and L. C. Baxevanaki. Statistical queries on historical relational databases. In Gergatsoulis and Rondogiannis [33], pages 229–243.

[84] Nikolaos S. Papaspyrou and Ioannis T. Kassios. GLU# embedded in C++: a marriage between multidimensional and object-oriented programming. *Softw., Pract. Exper.*, 34(7): 609–630, 2004. ISSN 0038-0644. doi: 10.1002/spe.582.

[85] Joey Paquet. Relational databases as multidimensional dataflows. Master's thesis, Departement d'Informatique, Université Laval, Québec, Canada, 1995.

[86] Joey Paquet. *Scientific Intensional Programming*. PhD thesis, Department of Computer Science, Laval University, Sainte-Foy, Canada, 1999.

[87] Joey Paquet. Distributed eductive execution of hybrid intensional programs. In *Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC'09)*, pages 218–224, Seattle, Washington, USA, July 2009. IEEE Computer Society. ISBN 978-0-7695-3726-9.

[88] Joey Paquet and Peter Kropf. The GIPSY architecture. In *Proceedings of Distributed Computing on the Web*, Quebec City, Canada, 2000.

[89] Joey Paquet and John Plaice. On the design of an indexical query language. In *Proceedings of the Seventh International Symposium on Lucid and Intensional Programming*, pages 28–36, 1994.

[90] Joey Paquet and John Plaice. The intensional relation. In Orgun and Ashcroft [80], pages 214–227.

[91] Joey Paquet and John Plaice. The semantics of dimensions as values. In Gergatsoulis and Rondogiannis [33], pages 259–273.

[92] Joey Paquet and Ai Hua Wu. GIPSY – a platform for the investigation on intensional programming languages. In *Proceedings of the 2005 International Conference on Programming Languages and Compilers (PLC 2005)*, pages 8–14. CSREA Press, June 2005.

[93] Joey Paquet, Serguei A. Mokhov, Emil I. Vassev, Xin Tong, Yi Ji, Amir H. Pourteymour, Kaiyu Wan, Aihua Wu, Sleiman Rabah, Bin Han, Bo Lu, Lei Tao, Yimin Ding, Chun Lei Ren, and The GIPSY Research and Development Group. The General Intensional Programming System (GIPSY) project. Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, 2002–2013. `http://newton.cs.concordia.ca/~gipsy/`, last viewed April 2012.

[94] Joey Paquet, Aihua Wu, and Peter Grogono. Towards a framework for the General Intensional Programming Compiler in the GIPSY. In *Proceedings of the 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2004)*, pages 164–165, New York, NY, USA, October 2004. ACM. doi: 10.1145/1028664.1028731.

[95] Joey Paquet, Serguei A. Mokhov, and Xin Tong. Design and implementation of context calculus in the GIPSY environment. In *Proceedings of the 32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC)*, pages 1278–1283, Turku, Finland, July 2008. IEEE Computer Society. doi: 10.1109/COMPSAC.2008.200.

[96] Manuel Peralta, Supratik Mukhopadhyay, and Ramesh Bharadwaj. Automatic synthesis and deployment of intensional kahn process networks. In Dominik Ślęzak, Tai hoon Kim, Stephen S. Yau, Osvaldo Gervasi, and Byeong-Ho Kang, editors, *Grid and Distributed Computing*, volume 63 of *Communications in Computer and Information Science*, pages 73–87. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-10548-7. doi: 10.1007/978-3-642-10549-4_10.

[97] John Plaice. Particle in-cell simulation with Lucid. In Orgun and Ashcroft [80], pages 149–161.

[98] John Plaice. Cartesian programming. Technical Report UNSW-CSE-TR-1101, University of Grenoble, France, January 2011. Habilitation Thesis, online at `ftp://ftp.cse.unsw.edu.au/pub/doc/papers/UNSW/1101.pdf`.

[99] John Plaice and Peter G. Kropf. Intensional communities. In Gergatsoulis and Rondogiannis [33], pages 292–295.

[100] John Plaice and William W. Wadge. A new approach to version control. *IEEE Transactions on Software*, 19(3):268–276, March 1993.

[101] John Plaice, Ridha Khedri, and Rene Lalement. From abstract time to real time. In *In Proceedings of the Sixth International Symposium on Lucid and Intensional Programming*, pages 83–93, 1993.

[102] John Plaice, Blanca Mancilla, Gabriel Ditu, and William W. Wadge. Sequential demand-driven evaluation of eager TransLucid. In *Proceedings of the 32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC)*, pages 1266–1271, Turku, Finland, July 2008. IEEE Computer Society. doi: 10.1109/COMPSAC.2008.191.

[103] Amir Hossein Pourteymour. Comparative study of Demand Migration Framework implementation using JMS and Jini. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, September 2008.

[104] Amir Hossein Pourteymour, Emil Vassev, and Joey Paquet. Towards a new demand-driven message-oriented middleware in GIPSY. In *Proceedings of PDPTA 2007*, pages 91–97. PDPTA, CSREA Press, June 2007.

[105] Amir Hossein Pourteymour, Emil Vassev, and Joey Paquet. Design and implementation of demand migration systems in GIPSY. In *Proceedings of PDPTA 2009*. CSREA Press, June 2008.

[106] Sleiman Rabah, Serguei A. Mokhov, and Joey Paquet. An interactive graph-based automation assistant: A case study to manage the GIPSY's distributed multi-tier runtime system. [online], June 2012. Submitted for publication to RACS'13; Pre-print: `http://arxiv.org/abs/1212.4123`.

[107] Alessandra Raffaetà and Thom Frühwirth. Two semantics for temporal annotated constraint logic. In Gergatsoulis and Rondogiannis [33], pages 78–92.

[108] Toby Rahilly and John Plaice. A multithreaded implementation for TransLucid. In *Proceedings of the 32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC)*, pages 1272–1277, Turku, Finland, July 2008. IEEE Computer Society. doi: 10.1109/COMPSAC.2008.191.

[109] Anand Ranganathan and Roy H. Campbell. A middleware for context-aware agents in ubiquitous computing environments. In Markus Endler and Douglas Schmidt, editors, *Proceedings of Middleware 2003*, volume 2672 of *Lecture Notes in Computer Science*, pages 143–161. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-40317-3. doi: 10.1007/3-540-44892-6_8.

[110] Chun Lei Ren. General intensional programming compiler (GIPC) in the GIPSY. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, 2002.

[111] Panagiotis Rondogiannis. *Higher-Order Functional Languages and Intensional Logic*. PhD thesis, Department of Computer Science, University of Victoria, Victoria, Canada, 1994.

[112] Panagiotis Rondogiannis. Adding multidimensionality to procedural programming languages. *Software: Practice and Experience*, 29(13):1201–1221, 1999. ISSN 1097-024X. doi: 10.1002/(SICI)1097-024X(199911)29:13⟨1201::AID-SPE278⟩3.0.CO;2-0.

[113] Panos Rondogiannis. Adding multidimensionality to procedural programming languages. In Gergatsoulis and Rondogiannis [33], pages 274–291.

[114] Panos Rondogiannis and William W. Wadge. Extending the intensionalization algorithm to a broader class of higher-order programs. In Orgun and Ashcroft [80], pages 228–233.

[115] M. C. Schraefel, Blanca Mancilla, and John Plaice. Intensional hypertext. In Gergatsoulis and Rondogiannis [33], pages 40–54.

[116] Paul Swoboda. *A Formalisation and Implementation of Distributed Intensional Programming*. PhD thesis, The University of New South Wales, Sydney, Australia, 2004.

[117] Paul Swoboda and John Plaice. An active functional intensional database. In F. Galindo, editor, *Advances in Pervasive Computing*, pages 56–65. Springer, 2004. LNCS 3180.

[118] Paul Swoboda and John Plaice. A new approach to distributed context-aware computing. In A. Ferscha, H. Hoertner, and G. Kotsis, editors, *Advances in Pervasive Computing*. Austrian Computer Society, 2004. ISBN 3-85403-176-9.

[119] Paul Swoboda and William W. Wadge. Vmake and ISE general tools for the intensionalization of software systems. In Gergatsoulis and Rondogiannis [33], pages 310–320. ISBN: 981-02-4095-3.

[120] Lei Tao. Warehouse and garbage collection in the GIPSY environment. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, 2004.

[121] S. Tao. *TLucid and Intensional Attribute Grammars*. PhD thesis, Department of Computer Science, Victoria University, Canada, 1994.

[122] Xin Tong. Design and implementation of context calculus in the GIPSY. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, April 2008.

[123] Xin Tong, Joey Paquet, and Serguei A. Mokhov. Complete context calculus design and implementation in GIPSY. [online], 2007–2008. `http://arxiv.org/abs/1002.4392`.

[124] J. van Benthem. *A Manual of Intensional Logic*. CSLI Publications, Stanford and The University of Chicago Press, 1988. ISBN 0-937073-29-6.

[125] Emil Vassev and Joey Paquet. A general architecture for demand migration in a demand-driven execution engine in a heterogeneous and distributed environment. In *Proceedings of the 3rd Annual Communication Networks and Services Research Conference (CNSR 2005)*, pages 176–182. IEEE Computer Society, May 2005. doi: 10.1109/CNSR.2005.9.

[126] Emil Vassev and Joey Paquet. A generic framework for migrating demands in the GIPSY's demand-driven execution engine. In *Proceedings of the 2005 International Conference on Programming Languages and Compilers (PLC 2005)*, pages 29–35. CSREA Press, June 2005.

[127] Emil Vassev and Joey Paquet. Towards autonomic GIPSY. In *Proceedings of the Fifth IEEE Workshop on Engineering of Autonomic and Autonomous Systems (EASE 2008)*, pages 25–34. IEEE Computer Society, 2008. ISBN 978-0-7695-3140-3. doi: 10.1109/EASe.2008.9.

[128] Emil Iordanov Vassev. General architecture for demand migration in the GIPSY demand-driven execution engine. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, June 2005. ISBN 0494102969.

[129] Lorenzo Verdoscia. ALFA fine grain dataflow machine. In Orgun and Ashcroft [80], pages 110–134.

[130] William W. Wadge. Possible WOOrlds. In Orgun and Ashcroft [80], pages 56–62. Invited Contribution.

[131] William W. Wadge. Intensional logic in context. In Gergatsoulis and Rondogiannis [33], pages 1–13. Tutorial.

[132] William W. Wadge. Hammings problem example. [online], December 2003. `http://i.csc.uvic.ca/home/hei/lup/contents.html`.

[133] William W. Wadge and Edward A. Ashcroft. *Lucid, the Dataflow Programming Language*. Academic Press, London, 1985.

[134] William W. Wadge and M. C. Schraefel. Putting the hyper back in hypertext. In Gergatsoulis and Rondogiannis [33], pages 31–39.

[135] William W. Wadge and Alan Yoder. The Possible-World Wide Web. In Orgun and Ashcroft [80], pages 207–213.

[136] William W. Wadge, G. Brown, M. C. Schraefel, and T. Yildirim. Intensional HTML. In *4th International Workshop PODDP'98*, March 1998.

[137] Kaiyu Wan. *Lucx: Lucid Enriched with Context*. PhD thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, 2006.

[138] Kaiyu Wan, Vasu Alagar, and Joey Paquet. A context theory for intensional programming. In *Workshop on Context Representation and Reasoning (CRR05)*, July 2005.

[139] Kaiyu Wan, Vasu Alagar, and Joey Paquet. Lucx: Lucid enriched with context. In *Proceedings of the 2005 International Conference on Programming Languages and Compilers (PLC 2005)*, pages 48–14. CSREA Press, June 2005.

[140] Kaiyu Wan, Vasu Alagar, and Joey Paquet. Real time reactive programming in lucid enriched with contexts. In Zhiming Liu and Keijiro Araki, editors, *Theoretical Aspects of Computing - ICTAC 2004*, volume 3407 of *Lecture Notes in Computer Science*, pages 387–402. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-25304-4. doi: 10.1007/978-3-540-31862-0_28.

[141] Ai Hua Wu. Semantic checking and translation in the GIPSY. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, 2002.

[142] Ai Hua Wu. *OO-IP Hybrid Language Design and a Framework Approach to the GIPC*. PhD thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, 2009.

[143] Ai Hua Wu and Joey Paquet. Object-oriented intensional programming in the GIPSY: Preliminary investigations. In *Proceedings of the 2005 International Conference on Programming Languages and Compilers (PLC 2005)*, pages 43–47. CSREA Press, June 2005.

[144] Ai Hua Wu, Joey Paquet, and Peter Grogono. Design of a compiler framework in the GIPSY system. In *Proceedings of the 15th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2003)*, volume 1, pages 320–328. International Association of Science and Technology for Development, November 2003.

[145] Aihua Wu, Joey Paquet, and Serguei A. Mokhov. Object-oriented intensional programming: Intensional Java/Lucid classes. In *Proceedings of SERA 2010*, pages 158–167. IEEE Computer Society, 2010. ISBN 978-0-7695-4075-7. doi: 10.1109/SERA.2010.29. Online at: `http://arxiv.org/abs/0909.0764`.

[146] Satoshi Yamane. Real-time object-oriented specification and verification. In Orgun and Ashcroft [80], pages 162–185.

[147] Yu Zhang and Kang Zhang. Associative query for multi-version web documents. In Gergatsoulis and Rondogiannis [33], pages 55–64.

[148] Q. Zhao. Implementation of an object-oriented intensional programming system. Master's thesis, Department of Computer Science, University of New Brunswick, Canada, 1997.

Index

API
  [], 28
  bool, 26, 28
  bool::true, 28
  boolean, 26, 28
  byte, 28
  char, 28
  CodeSegment, 23
  Dimension, 28, 30
  dimension, 26, 28
  double, 26, 28
  Executor, 28
  Float, 28
  float, 26, 28
  GIPC, 23
  GIPSYArray, 28
  GIPSYBoolean, 28
  GIPSYCharacter, 28
  GIPSYContext, 30
  GIPSYDouble, 28
  GIPSYEmbed, 28
  GIPSYFloat, 28
  GIPSYFunction, 28, 30
  GIPSYIdentifier, 28
  GIPSYInteger, 28
  GIPSYObject, 28
  GIPSYOperator, 28, 30
  GIPSYString, 28
  GIPSYVoid, 28
  int, 26, 28
  java.lang, 28
  Long, 28
  long, 26, 28
  Method, 28
  Object, 28
  Preprocessor, 23, 28
  SemanticAnalyzer, 28
  SequentialThreadGenerator, 28
  special, 21
  String, 26, 28
  string, 26, 28
  StringBuffer, 26
  toString(), 28
  true, 26
  type, 21
  void, 26, 28
AST, 5

C, 5
C++, 6, 26

data types
  matching Lucid and Java, 26

Forensic Lucid, 2, 4, 20
Fortran, 5, 26
Frameworks
  GEE, 5
  GICF, 26
  GIPC, 5
  GIPSY Type System, 26

GEE, 5
GICF, 26
GIPC, 5
  Preprocessor, 23
GIPL, 1, 3–5, 19–21
GIPSY, 5, 6, 23, 26, 28
  Type System, 26
  Types, 26
GIPSY Program
  Segments, 23
GIPSY Type System, 26
GLU, 4–6
GLU#, 4, 6

Indexical Lucid, 4, 5

Java, 5, 23, 26, 28
JLucid, 4, 5, 26
JOOIP, 4, 26

Lucid, 1–6, 26, 28
  Family, 5
  History, 5
  Pipelined Dataflows, 5
Lucx, 2–5, 26
Lustre, 4

MARFL, 2–4, 20

Onyx, 4, 6

Partial Lucid, 4, 5
Perl, 26
Preprocessor, 23
    GIPC, 23
Python, 26

Segments
    #<IMPERATIVELANG>, 26
    #<INTENSIONALLANG>, 26
    #CPP, 26
    #FORENSICLUCID, 26
    #FORTRAN, 26
    #GIPL, 26
    #INDEXICALLUCID, 26
    #JAVA, 26
    #JLUCID, 26
    #JOOIP, 26
    #LUCX, 26
    #OBJECTIVELUCID, 26
    #ONYX, 26
    #PERL, 26
    #PYTHON, 26
    #TENSORLUCID, 26
    #TRANSLUCID, 26
    #funcdecl, 24, 30
    #typedecl, 24, 30

Tensor Lucid, 4, 5, 25, 26
Tools
    libintense, 6
TransLucid, 1, 4, 6, 19–21, 23
Types, 26