

EDEN: An Engine for Definitive Notations Design, Implementation and Evaluation

by
Edward Yun Wai Yung

A thesis submitted for the award of
a Master's degree by research in Computer Science

Department of Computer Science
University of Warwick
Coventry CV4 7AL
U.K.

In 2 volumes:

Volume I: MSc. Thesis
Volume II: Appendices

18/9/89

Volume I
MSc. Thesis

EDEN: An Engine for Definitive Notations Design, Implementation and Evaluation

by

Edward Yun Wai Yung

Supervisor: Dr. W. M. Beynon

Department of Computer Science
University of Warwick
Coventry CV4 7AL
U.K.

A thesis submitted for the award of
a Master's degree by research in Computer Science

Abstract

This thesis concerns the ideas and experiences of programming in EDEN which is an experimental language designed to be a general-purpose definitive (definition-based) language with some imperative features. EDEN supports a very simple form of definition—an uni-directional relationship among variables. In this thesis, we present the fundamental philosophy of the design of this language and the methods of implementing such an interpreter. We investigate the advantages and draw-backs of the current definitive approach by using EDEN as a prototyping language; we describe how a subset of DoNaLD—a definitive notation for line drawing—is implemented using the EDEN interpreter as the underlying computing engine. We compare the approach of definitive programming with those of other programming paradigms, including functional programming and constraint-based programming. The possibility of using definitive languages for software specification is investigated. We also explore the potential for developing definitive programming into a concurrent programming paradigm. Finally, we propose some possible ways of improving the current design of EDEN through introducing other programming concepts, such as object-oriented programming techniques.

Keywords: CAD, Constraint, Declarative languages, Definition, Definitive (definition-based) programming, Dependency maintainer, Formula, Functional/applicative languages, Graphics, Parallel computation, Procedural/imperative languages, Software specification.

Table of Contents

Volume I: MSc. Thesis

Table of Figures	iv
Table of Listings	v
Acknowledgements	vi
1 Introduction	1
1.1 Introduction of Definitive Paradigm	1
1.2 Examples of Definitive Systems	4
1.3 Advantages of the Definitive Paradigm	6
1.4 Motivation of the Design of EDEN	6
1.5 Methodologies of Implementing Definition Manager	9
1.6 Application of EDEN	10
2 Comparison of Programming Paradigms	13
2.1 Machine States	13
2.1.1 Procedural Languages	13
2.1.2 Functional Languages	14
2.1.3 Definitive Languages	16
2.2 The Human/Computer Interface	17
2.3 Definitive Programming versus Functional Programming	19
2.4 Definitive Programming versus Logic Programming	21
2.5 Other Related Works	22
2.5.1 Implementation Techniques	22
2.5.2 Constraints	23
2.5.3 Data Dependency and Qualitative Reasoning	24
2.6 Conclusion	25
3 Design of EDEN	26
3.1 Fundamental Philosophy	26
3.1.1 Formula Definition	26
3.1.2 Action Specification	28
3.1.3 Dependency Graph	30

3.1.4	Delay Evaluation and Execution of Definitions and Actions	33
3.1.5	Other Features	33
3.2	The EDEN System	34
3.2.1	Read/Write Variables	35
3.2.2	Formula Variables	35
3.2.3	Action Specification	37
3.2.4	Function	38
3.2.5	Data Types	39
3.2.6	Operators	40
3.2.7	Statements	40
3.3	Comparison of the Definition and Action Concepts	41
3.3.1	Emulate Definition using Action	42
3.3.2	Emulate Action using Definition	42
3.3.3	Discussion	43
4	EDEN Programming	46
4.1	A Window Specification Example	47
4.1.1	EDEN Style	49
4.1.2	Procedural Style	50
4.1.3	Object-Oriented Style	50
4.1.4	Discussion	52
4.2	Constraint Maintenance	52
4.2.1	Method I — "terminated by comparison"	53
4.2.2	Method II — "terminated by status"	55
4.2.3	Method III — "constraint operator templates"	57
4.2.4	Discussion	60
4.3	Simulation of Parallel Data Flow Machine	61
5	Implementation of EDEN Interpreter	67
5.1	Definitions As Functions	67
5.2	Data Dependency	68
5.3	Data Template	70
5.4	Implementation	73
5.4.1	Lexical Analyzer	74
5.4.2	Symbol Table	74
5.4.3	Parser	74
5.4.4	Intermediate Executable Code Generator	74
5.4.5	Definition Manager/Code Interpreter	75
5.5	Definition Management Scheme	75
5.5.1	Depth First Scheme	76

5.5.2 Breadth First Scheme	77
5.5.3 Comparison of The Two Schemes	79
6 Implementation of DoNaLD	81
6.1 Definitive Languages for Graphics	83
6.2 Experimental Implementation of DoNaLD	83
6.2.1 DoNaLD To EDEN Front End	85
6.2.2 EDEN Interpreter Back End	87
6.2.3 SunCore Graphics Package	87
6.3 Towards CAD	88
6.4 Testing	89
6.5 Conclusion	90
6.6 Related Works	91
7 Parallel Definitive Systems	93
7.1 Parallel Computer Architectures	93
7.2 Concurrent Programming Languages	94
7.3 Parallelism in Definitive Languages	95
7.4 Hard-wired Definitive System	97
7.5 Problems of Automatic Parallelism Detection	98
7.6 Related Works	99
7.7 Conclusion	100
8 Future Research	101
8.1 Object-oriented EDEN	101
8.2 Higher-order Definition	103
8.3 Macro Definitions	104
8.4 Input Interface Gap	106
8.5 Re-definition within Action Body	107
8.6 Low-level Definitive Machine	108
9 Conclusion	110
Bibliography and References	114

Volume II: Appendices

Appendix A: EDEN Language Guide
Appendix B: Celsius-Fahrenheit Convertor
Appendix C: Simulation of Parallel Data Flow Machine
Appendix D: Listing of the DoNaLD-to-EDEN Translator
Appendix E: Listing of the EDEN Interpreter

Table of Figures

Figure 3-1	An informal description of some objects	31
Figure 3-2	The dependency graph of objects described in Figure 3-1	31
Figure 3-3	The trigger propagation in the dependency graph	32
Figure 3-4	The block diagram of a formula maintaining sub-system	36
Figure 3-5	The block diagram of the action execution management sub-system	37
Figure 4-1	Abstract diagram of "×" and "+" constraint operator	58
Figure 4-2	Abstract diagram of a connector operator	58
Figure 4-3	Abstract diagram of the Celsius-Fahrenheit Converter constructed using constraint operators	59
Figure 4-4	A pipeline arrangement of processors for evaluating the polynomial $f(x,a,b,c,d) = a \cdot x^3 + b \cdot x^2 + c \cdot x + d$.	62
Figure 4-5	Diagram of a Processor Template	63
Figure 5-1	A trace of the definition manager scheme	72
Figure 5-2	The dependency graph of a set of definitions	77
Figure 5-3	The trace of the Depth First Scheme	78
Figure 5-4	The trace of the Breadth First Scheme	78
Figure 7-1	A typical shared-memory multiprocessor system	93
Figure 7-2	A typical multicomputer system	94
Figure 7-3	The arrangement of processors of the parallel computation of $f = ax^3 + bx^2 + cx + d$	96
Figure 7-4	The arrangement of a data pipeline derived from Figure 7-3	98

Table of Listings

Listing 4-1	A simple Celsius-Fahrenheit constraint (terminated by comparison)	54
Listing 4-2	A Celsius-Fahrenheit constraint (terminated by status)	56
Listing 4-3	General form of constraint actions where each action affects one variable only	56
Listing 4-4	A sample program of a connector operator	59
Listing 4-5	Main Program of the Celsius-Fahrenheit Converter constructed by basic constraint templates	60
Listing 4-6	An EDEN program of a data pipeline of solving the polynomial $f(x, a, b, c, d) = a \cdot x^3 + b \cdot x^2 + c \cdot x + d$	65

Acknowledgements

I wish to express my gratitude to my friends and colleagues who helped me and encouraged me during these years. In particular, many thanks to Dr. Steve Russ and Mike Slade for their useful comments.

Above all, I am most grateful to my supervisor, Dr. Meurig Beynon, for the continuous help, encouragement, and constructive criticisms he has given me during the work.

Finally, I wish to thank my family in Hong Kong for the love and moral support they showed during my stay in U.K.