

CONTENTS

1. Introduction	1
1.1. Overview	1
1.2. Bindings and Binding Times	1
1.3. Organization of This Volume	3
2. Examples from TEMPO and Some Current Programming Languages	5
2.1. A Simple Algorithm Expressed in Seven Different Languages	5
2.2. Some Features of TEMPO	16
2.2.1. Dynamic Data Structures	16
2.2.2. Symbolic Indirect Addressing	17
2.2.3. Dynamic Generation of Program Text	17
2.2.4. Procedure Parameter Substitution	18
3. Syntax of TEMPO	20
4. Semantics of TEMPO	23
4.1. Introduction and Informal Overview of TEMPO Semantics	23
4.2. Values of Variables	25
4.3. Snapshots and Segments	28
4.4. The Abstract Interpreter	34
4.4.1. Utility Routines	37
4.4.2. Routines to Handle Blocks and Scopes of Names	38
4.4.3. Expression Evaluation and Assignment	39
4.4.4. The IF Statement	42
4.4.5. The Goto Statement	43
4.4.6. Procedure Call and Return	44
5. Implementation Techniques for TEMPO	46
5.1. Semantics Versus Implementation	46
5.2. Linked Lists	47

5.3. The TEMPO Implementation Data Structures	48
5.4. The Program List	48
6. Machine Efficiency & Programmer Convenience	53
6.1. The Extremes--TEMPO versus FORTRAN	53
6.2. Trading Machine Efficiency for Programmer Convenience (and Vice Versa)	55
6.3. Sources of Inefficiency in TEMPO	56
7. Improvements to Increase Machine Efficiency	59
7.1. Overview	59
7.2. Storage Allocation	59
7.3. Creation and Manipulation of Program Text	67
7.4. Variable Names and Labels in the Snapshot	70
7.5. Data Types	77
7.6. Conditions for Compilability	80
8. Parameter Passing and Reference Variables	84
8.1. Procedures and Parameters	84
8.2. Reference Variables and Operations	84
8.3. Methods of Parameter Passing and Their Relative Efficiencies	88
8.4. Comparison of the Six Methods of Parameter Passing	93
8.5. The Dangling Reference Problem	94
9. Binding Times in Some Current Programming Languages	96
9.1. Introduction	96
9.2. Languages Designed for Efficient Execution: FORTRAN, COBOL, ALGOL 60, PASCAL	96
9.3. Multipurpose Languages: PL/I, ALGOL 68	97
9.4. Languages Designed for Programmer Convenience: APL, LISP, SNOBOL	98
9.5. Summary	100
10. Conclusions	103
10.1. Summary	103

10.2. Implications for the Design of Programming Languages	104
10.3. Further Topics in Programming Languages	106
Appendix A. Extended Backus-Naur Form Syntax Notation	108
Appendix B. TEMPO/SP - A Syntactically-Enriched Version of TEMPO for Structured Programming	111
References	116