Karlsson, Roland [1992]. A high-performance or-parallel Prolog system, Ph.D. diss., TRITA-TCS-LPS-9202, Department of Telecommunication and Computer Systems, The Royal Institute of Technology, Stockholm, and SICS Dissertation Series 07, Swedish Institute of Computer Science.

Keisu, Torbjörn [1994]. Tree Constraints, Draft Ph.D. diss., Department of Teleinformatics, The Royal Institute of Technology, Stockholm, and Swedish Institute of Computer Science.

Lassez, Jean-Louis, Michael J. Maher, and Kimbal G. Marriott [1988]. Unification revisited. In *Foundations of Deductive Databases and Logic Programming*. Morgan-Kauffman.

Lloyd, John W. [1989]. *Foundations of Logic Programming*. 3d ed. Springer-Verlag.

Lusk, Ewing, R. Butler, T. Disz, R. Olson, R. Overbeek, R. Stevens, D. H. D. Warren, A. Calderwood, P. Szeredi, S. Haridi, P. Brand, M. Carlsson, A. Ciepilewski, B. Hausman [1988]. The Aurora or-parallel Prolog system. In *Proceedings of the International Conference on Fifth Generation Computer Systems 1988*. Ohmsha, Ltd., Japan. (Also in [Carlsson 1990].)

Maher, Michael J. [1987]. Logic semantics for a class of committed choice programs. In *Logic Programming: Proceedings of the Fourth International Conference*. The MIT Press.

Maher, Michael J. [1988]. Complete axiomatizations of the algebra of finite, rational and infinite trees, Technical Report, IBM Thomas J. Watson Research Center.

Mariën, André and Bart Demoen [1991]. A New Scheme for Unification in the WAM. In *Logic Programming: Proceedings of the 1991 International Symposium*. The MIT Press.

Meier, Micha [1991]. Recursion vs. iteration in Prolog. In *Logic Programming: Proceedings of the Eighth International Conference*. The MIT Press.

Miyazaki, Toshihiko, Akikazu Takeuchi, and Takashi Chikayama [1985]. A sequential implementation of Concurrent Prolog based on the shallow binding scheme. In *1985 Symposium on Logic Programming*. IEEE Computer Society Press.

Montelius, Johan, and Khayri M. Ali [1994]. An and/or-parallel abstract machine for AKL (extended abstract). In Parallel Logic Programming and its Programming Environments, Technical Report CIS-TR-94-04, University of Oregon.

Moolenaar, Remco and Bart Demoen [1993]. A parallel implementation for AKL, In *Programming Language Implementation and Logic Programming* (*PLILP '93*). Springer-Verlag.

Moolenaar, Remco and Bart Demoen [1994]. Hybrid tree search in the Andorra model. In *Logic Programming: Proceedings of the Eleventh International Conference*. The MIT Press. (To appear.)

Naish, Lee [1988]. Parallelizing NU-Prolog. In *Logic Programming: Proceedings of the Fifth International Conference and Symposium*. The MIT Press.

O'Keefe, Richard A. [1990]. *The Craft of Prolog*. The MIT Press.

Palmer, Doug and Lee Naish [1991]. NUA-Prolog: an extension to the WAM for parallel Andorra. In *Logic Programming: Proceedings of the Eighth International Conference*. The MIT Press.

Plotkin, Gordon D. [1981]. A structural approach to operational semantics, DAIMI FN-19, Computer Science Department, Aarhus University.

Santos Costa, Vitor, David H. D. Warren, Rong Yang [1991a]. The Andorra-I engine: a parallel implementation of the Basic Andorra Model. In *Logic Programming: Proceedings of the Eighth International Conference*. The MIT Press.

Santos Costa, Vitor, David H. D. Warren, and Rong Yang [1991b]. The Andorra-I preprocessor: supporting full Prolog on the Basic Andorra Model. In *Logic Programming: Proceedings of the Eighth International Conference*. The MIT Press.

Saraswat, Vijay A. [1987a]. The concurrent logic programming language CP: definition and operational semantics. In *Conference Record of the Fourteenth Annual ACM Symposium on Principles of Programming Languages*. ACM Press.

Saraswat, Vijay A. [1987b]. CP as a general-purpose constraint language. In *Proceedings of the National Conference on Artificial Intelligence*. American Association of Artificial Intelligence.

Saraswat, Vijay A. [1987c]. The language GHC: operational semantics and comparison with CP($\downarrow$, |). In *Proceedings 1987 Symposium on Logic Programming*. IEEE Computer Society Press.

Saraswat, Vijay A. [1989]. Concurrent constraint programming languages, Ph.D. diss., Carnegie-Mellon University. (Also available in Doctoral Dissertation Award and Logic Programming Series, MIT Press, 1993.)

Saraswat, Vijay A. and Martin Rinard [1990]. Concurrent constraint programming. In *Conference Record of the Seventh Annual ACM Symposium on Principles of Programming Languages*. ACM Press.

Saraswat, Vijay A., Martin Rinard, and Prakash Panangaden [1991]. Semantic foundation of concurrent constraint programming. In *Conference Record of the Eighteenth Annual ACM Symposium on Principles of Programming Languages*. ACM Press.

Schulte, Christian, and Gert Smolka [1994]. Encapsulated search in Oz, Draft Research Report, DFKI.

Shapiro, Ehud [1983]. A subset of Concurrent Prolog and its interpreter, Technical Report TR-003, ICOT. (Revised version in [Shapiro 1987].)

Shapiro, Ehud and Akikazu Takeuchi [1983]. Object-oriented programming in Concurrent Prolog. *Journal of New Generation Computing* **1**(1). (Revised version in [Shapiro 1987]).

Shapiro, Ehud and Colin Mierowsky [1984]. Fair, biased, and self-balancing merge-operators: their specification and implementation in Concurrent Prolog. *Journal of New Generation Computing* **2**(3). (Also in [Shapiro 1987].)

Shapiro, Ehud [1986a]. Concurrent Prolog: a progress report. *IEEE Computer* **8**(19). (Also in [Shapiro 1987].)

Shapiro, Ehud [1986b]. A test for the adequacy of a language for an architecture, Technical Report CS86-01, Weizmann Institute of Science. (Revised version in [Shapiro 1987].)

Shapiro, Ehud and Shmuel Safra [1986]. Multiway merge with constant delay in Concurrent Prolog. *Journal of New Generation Computing* **4**(3). (Revised version in [Shapiro 1987].)

Shapiro, Ehud [1987], ed. *Concurrent Prolog: Collected Papers*. 2 vols. The MIT Press.

Shapiro, Ehud [1989]. The family of concurrent logic programming languages. *ACM Computing Surveys* **21**(3):412–510.

Shriver, Bruce and Peter Wegner [1987], eds. *Research Directions in Object-Oriented Programming*. The MIT Press.

Smolka, Gert and Ralf Treinen [1992]. Records for logic programming. In *Logic Programming: Proceedings of the Joint International Conference and Symposium on Logic Programming*. The MIT Press.

Smolka, Gert [1993]. Residuation and guarded rules for constraint logic programming. In *Constraint Logic Programming: Selected Research*. The MIT Press.

Smolka, Gert, Martin Henz, and Jörg Würtz [1993]. Object-oriented concurrent constraint programming in Oz, Research Report RR-93-16, German Research Center for Artificial Intelligence (DFKI), Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany.

Smolka, Gert [1994]. A calculus for higher-order concurrent constraint programming with deep guards, Research Report RR-94-03, German Research Center for Artificial Intelligence (DFKI), Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany.

Smolka, Gert, et al. [1994]. The Oz handbook, Research Report, German Research Center for Artificial Intelligence (DFKI), Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany.

Snyder, A., W. Hill and W. A. Olthoss [1989]. A glossary of common object-oriented terminology, STL-89-26, Software Technology Laboratory, Hewlett-Packard Laboratories.

Sterling, Leon and Ehud Shapiro [1986]. *The Art of Prolog*. The MIT Press.

Taylor, Stephen [1989]. *Parallel logic programming techniques*. Eaglewood Cliffs.

Tick, Evan [1991]. *Parallel Logic Programming*. The MIT Press.

Tribble, Eric Dean, Mark S. Miller, Kenneth M. Kahn, Daniel G. Bobrow, Curtis Abbott, and Ehud Shapiro [1987]. Channels: a generalisation of streams. In *Logic Programming: Proceedings of the Fourth International Conference*. The MIT Press. (Also in [Shapiro 1987].)

Ueda, Kazunori [1985]. Guarded Horn Clauses, TR-103, ICOT, Tokyo. (Revised version in [Shapiro 1987]).

Ueda, Kazunori and Takashi Chikayama [1990]. Design of the kernel language for the parallel inference machine. *The Computer Journal* **33**(6).

Ueda, Kazunori and Masao Morita [1992]. Message-oriented parallel implementation of moded Flat GHC. In *Fifth Generation Computer Systems 1992*. IOS Press.

Van Hentenryck, Pascal [1989]. *Constraint Satisfaction in Logic Programming*. The MIT Press.

Van Hentenryck, Pascal and Yves Deville [1991]. The cardinality operator: a new logical connective and its application to constraint logic programming. In *Logic Programming: Proceedings of the Eighth International Conference*. The MIT Press.

Van Hentenryck, Pascal, Vijay Saraswat, and Yves Deville [1992]. Constraint logic programming over finite domains: the design, implementation, and applications of cc(FD), Technical report, Computer Science Department, Brown University.

Wadler, Philip [1990]. Comprehending monads. In *ACM Conference on Lisp and Functional Programming*. ACM Press.

Warren, David H. D. [1982]. Higher-order extensions to Prolog: are they needed? In *Machine Intelligence 10*, Ellis Horwood with John Wiley and sons, Lecture Notes in Mathematics 125.

Warren, David H. D. [1983]. An abstract Prolog instruction set, Technical Note #309, Artificial Intelligence Center, SRI International.

Warren, David H. D. [1989]. The Extended Andorra Model with implicit control. Presentation at the ICLP'90 Preconference Workshop on Parallel Logic Programming in Eilat, Israel. Workshop record available from SICS, Box 1263, S-164 28 KISTA, Sweden.

Yang, Rong [1987]. *P-Prolog – A Parallel Logic Programming Language*. Series in Computer Science, Vol. 9, World Scientific.

Yang, Rong [1989]. Solving simple substitution ciphers in Andorra-I. In *Logic Programming*: *Proceedings of Sixth International Conference*. The MIT Press.

Yoshida, Kaoru and Takashi Chikayama [1988]. A'UM – A stream-based concurrent object-oriented language. In *Proceedings of the International Conference on Fifth Generation Computer Systems 1988*. Ohmsha, Ltd., Japan.

INDEX

A

A’UM 120
a-registers 84
abstract machine,
  for AKL 80-119
Actors 145
actual parameter 10
agent 8, 10
agent definition 10, 46
AGENTS 23, 80, 138
aggregate 46
aggregate box 48
aggregate rule 52
aggregate execution rule 70
aggregates, incremental 117
AKL
  computation model 56
  goal transition system 49
  informal introduction 7-23
all-solution group, cc combinator 185
alternatively, pragma in KL1 172
and-box 48
  solved 48
and-box suspension rule 72
and-box task 65
and-continuation 83
and-instructions 94
and-node 81
and-parallel composition,
  cc combinator 186
and-parallelism 166
and-stack 85
and-task 65
Andorra 168
Andorra-I 168
answers 56
arithmetic expression 9
arity of atom 10, 45
ask 8
ask, cc combinator 184
at-least-one constraint 39
at-most-one constraint 39
atomic tell, cc combinator 184
authoritative guard 59

B

back up from and-node 89
bagof statement 20
  ordered 20
  unordered 20
bags 131
Basic Andorra Model (BAM) 168
bind variable 88
binding 9
body
  of clause 14, 15, 17
  of definition 46
bound variable 46
box
  aggregate 48
  and 48
  choice 48
  current 64
  or 48

C

call 10
call agent 136
  n-ary 147
candidate for choice splitting 51
candidate clause 155
candidate, selecting 92
cardinality operator 179
cc combinator
  all-solution group 185
  and-parallel composition 186
  ask 184
  atomic tell 184
  eventual tell 184
  existential quantification 186
  indeterminate prefixing 185
  nondeterministic prefixing 184
  or-selection 186
  reconciliation 185
  single-solution group 185
  synthesize 185
cc framework 61, 183-187
channels 130
choice 46
  committed 14
  conditional 10, 17
  cut 57
  logical conditional 57
  nondeterminate 15
choice execution rule 70
choice instructions 94
choice rule 50
choice splitting execution rule 73
choice splitting rule 51
choice splitting task 65, 84
choice splitting, attempt 92
choice-box 48
choice-box suspension rule 73
choice-continuation 84
choice-node 82
choice-stack 85
choice-task 65
chunk 98
class 30
clause 46
  guarded 14, 15, 17
  syntax 21
clause composition, parallel 172
clause composition, sequential 172
clause rule 70
clause task 65
clausewise reduction rule 154
close
  medium 123
  port 131
CLP 7, 153
code generation 97-107
code, editing for efficiency 104
code, examples of 104
code, for aggregates 102
code, for atoms 99
code, for choice 101
code, for clauses 101
code, for composition 98
code, for initial statements 103
collect and-node 93
collect execution rule 71
collect goal 48
collect operation 46
collect rule 52
collect statement 52
commit execution rule 71
commit operator 14
commit rule 51
committed choice statement 14, 46
committed-choice languages 169-176
communication medium 123
complete computation 60
completed program 59
completeness 60
composition rule 50
composition execution rule 70
composition statement 10, 46
computation 56
  complete 60
  failed 56
  logical 59
  normal 60
computation model 4
  AKL 45, 56-62
computation rule
  aggregate 52
  choice 50
  choice splitting 51
  collect 52
  commit 51
  composition 50
  condition 50
  constraint atom 49
  constraint simplification 52
  environment failure 51
  goal failure 51

guard distribution 51
guard failure 51
hiding 50
logical condition 57
noisy cut 57
or-flattening 52
program atom 50
promotion 50
quiet cut 57
subgoal 49
unit 52
computation rules 49-58
aggregate 52
failure 51
pruning 50
quiet 50, 52
concurrency 4
concurrent constraint programming 8, 183
concurrent logic programming 7, 169
concurrent objects 136
concurrent programming 25
Concurrent Prolog 7, 169
condition execution rule 71
condition rule 50
conditional choice statement 10, 17, 46
configuration 55
final 56
initial 56
labelled 64
stuck 56
terminal 56
confluence 5, 60
confluent computations 60
conjunction, parallel 172
conjunction, sequential 172
constant 9
constant delay merger 125
constant delay property 123
constraint 8, 47, 83
at-least-one 39
at-most-one 39
exactly-one 38
false 10
propagation 38
true 10, 47
constraint atom 45
constraint atom rule 49
constraint atom failure rule 69
constraint atom success rule 69
constraint atom statement 10
constraint logic programming 7, 153, 176, 183
constraint name 45
constraint programming 38
constraint satisfaction problem 38
constraint simplification rules 52
constraint store 8
external 64
local 64
constraint theory 9, 47
constructor expression 9
consume message in medium 123
consumer 25
context 48, 85
of goal 49
of labelled configuration 64
context list 65
context stack 85
continuation, success 158
continuations, to messages 136
copy avoidance 115
copy, for choice splitting 92
copying, in abstract machine 111-116
correctness, of execution model 78
CP 61
current and-node register 86
current argument register 86
current box 64
current choice-node 86
current context 87
cut 159
noisy 57
soft 57
cut choice statement 57
cut operator 57

D

data areas 85-86
data objects, in abstract machine 81
dead and-node 87
dead goal 64
decode instructions 88
deep guard 17
defining model 4
definite clause programs 155
definition
agent 10
of agent 46
definitionwise reduction rule 155
dependent and-parallelism 166
deref 53
dereference and-node 87
dereference, register 89

dereference, tree 89
derivation 56
design 1-6, 7-8
  efficient 2
  expressive 2
  formal 3
  parallel execution 4
  simple 2
determinate and-node 87
determinate atom 168
determinate mode 49
determinate phase 168
disentailment 8
distributor 27
don't care nondeterminism 5, 14
don't know nondeterminism 15, 51
  encapsulation of 42
  scope of 18

E

eager waking 80, 116
empty list 9
encapsulation
  of don't know nondeterminism 42
  of object 31
engine 42
engines 138
entailment 8
environment 64
  of context 49
  of goal 49
  of local agent 18
environment failure rule 51
Erlang 149
eventual tell, cc combinator 184
exactly-one constraint 38
execution 63, 76
  multiple worker 64
  partial 76
  single worker 64
execution model 4, 69
  AKL 63
execution rule
  aggregate 70
  and-box suspension 72
  choice 70
  choice splitting 73
  choice-box suspension 73
  clause 70
  collect 71
  commit 71
  composition 70
  condition 71
  constraint atom failure 69
  constraint atom success 69
  failure in or 72
  goal failure 72
  guard failure 72
  hiding 70
  installation failure 76
  installation success 75
  or-box suspension 73
  program atom 69
  promote after splitting 75
  promotion 71, 72
  stable and-box detection 72
  unit 73
  wake-up 75
  woken-up 75
execution rules 69-76
execution state 69
  final 76
  initial 76
  terminal 76
execution, in abstract machine 85, 87-93
existential quantification,
  cc combinator 186
expression 9
  arithmetic 9
  constant 9
  constructor 9
  empty list 9
  list constructor 9
  number 9
  variable 9
Extended Andorra Model (EAM) 61
external constraint store 64
external variable 64, 87

F

fail and-node 89
failure 17
  of computation 56
  of local computation 18
failure in or execution rule 72
fairness, of execution model 78
false 10
feature trees 48
FGHC 171
final configuration 56
final execution state 76
finite trees 48
Flat GHC 171
flat guard 17
flat guard optimisation 107

formal parameter 10, 46
functional language 6
functional programming 34

G

garbage collection 54
  of objects 122
general goal 48
GHC 7, 170
  Flat 171
global goal 48
goal 48
  general 48
  global 48
  guarded 48
  labelled 64
  local 48
  well-formed 55
goal failure execution rule 72
goal failure rule 51
goal transition system 49
graph traversal example 142
guard 14, 15, 17
  authoritative 59
  indifferent 59
guard distribution rule 51
guard failure execution rule 72
guard failure rule 51
guard instructions 95
guard operator 46
guarded clause 14, 15, 17
guarded goal 48

H

Haskell 34
head
  of clause 21
  of definition 46
heap 85
hiding rule 50
hiding execution rule 70
hiding statement 11, 46
higher-order 35
histogramming example 141

I

Id 144
ignored variables for suspension 66
imperative language 5
impose task 65
incompatible constraints 47
incomplete messages 28
incremental aggregates 117
independent and-parallelism 166
indeterminate prefixing, cc combinator 185
indifferent guard 59
inheritance 6, 32
initial configuration 56
initial execution state 76
insertion point register 86
install and-node 91
install task 65
installation failure rule 76
installation success rule 75
instance 30
instruction decoding 88
instruction set 93-97
instructions, and 94
instructions, choice 94
instructions, guard 95
instructions, tree-constraint 96
instructions, unification 96
integration of paradigms 41
interfering sets of local variables 55
interoperability 5, 41

K

KAP 61
Kernel Andorra Prolog 61
Kernel PARLOG 172
KL1 171
knowledge information processing 1, 7

L

label of goal 64
  new 69
labelled configuration 64
labelled goal 64
lazy waking 80, 116
lifetime, of variable 103
Linda 146
Linear Logic 133, 187
Lisp 143
list comprehension 20
lists 9
live and-node 87

live goal 64
local constraint store 64
local execution 18, 177
local forking reduction rule 169
local goal 48
local search 181
local variable 64, 87
  of and-box 48
  of clause 14
logic programming 7
logical computation 59
logical condition rule 57
logical condition operator 57
logical conditional choice 57
logical interpretation
  of definitions 58
  of goals 58
  of statements 58
logical language 6

M

mailbox communication 145
merger 27, 124
  binary 27
  constant delay 125
  multiway 125
  multiway in AKL 127
message-oriented scheduling 118, 162
metainterpreter for side effects 164
metalogical operations 165
method of object 30
mode 49
  determinate 49
  nondeterminate 49
  simplification 49
model
  computation 4
  defining 4
  execution 4
monitor-controller 177, 178
monotonicity of constraint store 121, 123
moving, in configuration 64
MRB 139, 172
multiparadigm language 24
multiple worker execution 64
multiple writers 124, 126
multiway merger 125
  in AKL 127
mutual references 130

N

N-queens problem 38
natural-language processing 36
negation, as failure 162
new label 69
no-choice task 84
noisy constraint 47
noisy cut 57
noisy cut rule 57
noisy pruning 159
nondeterminate choice statement 15, 46
nondeterminate mode 49
nondeterminate phase 168
nondeterminism 5
  don't care 5, 14
  don't know 15, 51
  encapsulation of 42
nondeterministic prefixing,
  cc combinator 184
normal computation 60
normal program 97
NUA-Prolog 168
null reference 81
number 9

O

object 30
object-based language 6
object-oriented language 6
object-oriented programming 1, 29
objects
  as processes 120
  garbage collection of 122
open_port 131
optimisations, of abstract machine 107-111
or-box 48
or-box suspension rule 73
or-box task 65
or-component 57
or-fair 60
or-flattening rule 52
or-parallel Prolog 157
or-parallelism 166
or-selection, cc combinator 186
otherwise, pragma in KL1 172
Oz 62, 187-189