CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

Concept index

$\mathcal{A}$ (automatically generated function in identifier cross-reference), 39
abbreviations, type, 529
abstract classes, 622, 624, 656–662, 708
  `Collection`, 642
  `Integer`, 664
  `LargeInteger`, 669
  `Number`, 663
abstract data types, *see* abstract types
abstract machine(s), 30, 67
  further reading, 246
  Impcore, 30
  for implementation, 246
  μScheme, 144
  μScheme+, 211
  μSmalltalk, 678–679
  representations
    μSmalltalk, 686–687
  in semantics, 211
abstract-machine semantics, 244
abstract syntax, 14, 15, 67
  exceptions, 589
  Impcore, 27–28
  μML, 485–486
  μScheme (in ML), 306–307
  μScheme (in C), 144–145
  μScheme+, 225–226
  μScheme+ stack frames, 225–226
  μSmalltalk, 687–688
  nano-ML, 404–405
  Typed Impcore, 332–333
  Typed μScheme, 361
abstract-syntax trees, 27–28
  in C, 41–43
abstract types, 455, 525, 538, 585
  in C, 39
  as components of modules, 538
  design choices, 580–583
  equality and, 634
  examples, 526
  and exceptions, 583–584
  in Java, 707
  limitations, 585
  objects vs, 625–627
  proofs of correctness, 588
  in real languages, 580–583
abstraction(s), 527, 528, *see also* data abstraction
  immutable, 545
  mutable, 545
  syntactic, *see* macros
abstraction functions, 545, 549 (defined), 586, 708
  circular lists, 673
  complex numbers, 551
  data structures, typical and, 549–551
  dictionaries, 549
  examples, 552
  natural numbers, 670, 671
  priority queues, 550, 552
  sets, 548
  two-dimensional points, 551
abstraction, functional, 36
access paths, 561 (defined)
  absolute, 559
  environment lookup, 574
  of modules or components, 559
  pseudo-bindings, 563, 574
  relative, 575
accessor(s), *see* observers
accessor functions for records, 107
accumulating parameters, 99–101
ad hoc polymorphism, 132, 544, *see also* overloading
algebraic data types, 317, 501
  compile-time checks, 499–500
  extensions, 503
  generalized, 503
  in μML, 466–476
  in Molecule, 535
  origins, 502
  polymorphic, 462–463
  in real languages, 499–501
  recursive, 463–464
algebraic laws, 98
  association lists, 106, 112
  binary-tree nodes, 109
  in calculational proofs, 115
  case expressions, 476–477
  continuation-passing style and, 138

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

algebraic laws (continued)
currying and uncurrying, 126
equality of S-expressions, 104
finite maps, 112
further reading, 175
`has?` (μScheme function), 103
higher-order list functions, 130
`if` expressions, 112
and imperative features, 403
insertion sort, 101
length of a list, 98
length of appended lists, 115
list append, 99
list insertion, 101
list primitives, 111
list reversal, 100
membership in S-expressions, 103
notation for lists, condensed, 99
operator classifications and, 111–112
pair types, 349
product types, 349
proofs of, 114, 115
set functions, 105
simplifying code with, 113
solving conjunctive normal form, 141, 142
soundness, 65
in specifications, 547, 548
substitutions, 409–410
testable properties as, 110
tree-node functions, 109
tuple types, 349
uses, 175
algebraic specification, 547, 548
allocation, 259
in functional languages, 205
implementations, 162 (μScheme), 266 (μScheme+)
interfaces, 154
in μScheme+, 266
mutability and, 545
in procedural languages, 206
`allof`, 541, 552, 555
alpha-conversion (renaming)
type variables, 368–369
and ($\wedge$, type intersection), 564
`and`, meaning in ML, 303
answer(s), 708
to messages, 610 (defined)
answer (verb form)
in Smalltalk, 612
answer types
in continuation-passing style, 138
APIs, 527, 586
examples, 531–532
`append`, 99 (μScheme), 475 (μML)
proof of laws, 114
applicative languages, 402
`apply` primitive (exercise), 198
arguments, variable number of, 169
arithmetic, multiprecision, *see* multiprecision arithmetic
arity
μSmalltalk message names, 627
primitives, 313
arrays
associative, *see* finite maps
implementation (μSmalltalk), 661–662
semantics, 345
syntax, 343–345
type checking, 348
types for, 343
typing rules, 346–348
arrow types
typing rules, 348
ascriptions
of a module type to a module, 564
in Standard ML, 441
type checking, 564
assignment
to Boolean variables, 138
satisfying a Boolean formula, 138
assignment (syntactic form `set`)
implementations
Impcore, 49
μScheme, 156
μScheme+, 230
operational semantics
Impcore, 34
μScheme, 145
μSmalltalk, 680
small-step semantics, 217
typing rules
Typed Impcore, 335
Typed μScheme, 363
association lists, 105–107, 646, S73
algebraic laws, 106, 112
attributes, 105
continuations and, 136–137
equality, 132–133
in full Scheme, 169
keys, 105
sets of, 132–135
associative arrays, *see* finite maps
ASTs, *see* abstract-syntax trees
`@` (instantiation), 352, 358, 412
examples, 358
implicit, 413
typing rules, 365
atoms, 172, S95
in μScheme, 92
in Prolog, S43

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

autognostic principle, 626, 708
automatic instantiation, 409
automatic memory management, 288
  benefits, 257
  performance, 260

$\mathcal{B}$ (basis function in identifier cross-reference), 39
backtracking
  using continuation-passing style, 138–143
  visualizations, 139
Backus-Naur Form, 17
base types, 343
  Typed Impcore, 333
basis, 26 (defined), 67
  Impcore, 29
  initial, *see* initial basis
  Typed μScheme, 382
bearskins, *588*
`begin`
  operational semantics
    Impcore, 35
    μScheme, 149
    μSmalltalk, 681
  syntactic sugar, 167
  typing rules
    nano-ML, 413 (nondeterministic), 446 (constraint-based)
    Typed Impcore, 336
    Typed μScheme, 363
behavior(s), μSmalltalk
  representation, 687
  semantics, 678–679
behavioral specifications, 527
  examples, 529, 547
behavioral subtyping, 627, 708, 713
big-step semantics, *see* operational semantics
bignums, *see* large integers
binary operations
  on abstract types, 555–557
  on objects, 662–673
  primitives, embedded as, 313
binary search trees
  exercises, 508–509
  representation invariants, 550
binary trees
  algebraic laws, 109
  encoded as S-expressions, 109–110
  exercises, 508–509
  inference rules, to define, 109
  μScheme, 109–110
  traversals
    breadth-first, 118–120
    inorder, 181
    level-order, 118–120
    postorder, 181
binary trees (continued)
  traversals (continued)
    preorder, 110
binding(s)
  access paths, 563, 574
  in Molecule type-checking environments, 563
binding occurrences
  type variables, 371 (defined)
black magic, decision procedure for, *S45*
black objects, 261
  in copying collection, 273
  examples, 275
  in mark-sweep collection, 269
blocks (μSmalltalk), 618, 627–629, 708
  bootstrapping, 698
  closures and, 618
  evaluation, 691
  operational semantics, 681
  protocol, 630, 638–641
BNF, 17
Boolean(s)
  Church encoding, 655
  μSmalltalk, 630
    bootstrapping, 697–699
`boolean?`, 92
Boolean classes
  implementations, 655
  protocol, 639
Boolean operators, short-circuit, 164–165
Boolean satisfiability, 138–139
  solver, 139–143
bootstrapping, μSmalltalk
  blocks, Booleans, literals, 698–699
bound occurrences
  of type variables, 371
bound type variables, 371 (defined)
bound variables, 315–317
brackets
  square vs round, 117
  as tokens, 17
breadth-first traversals, 118–120
`break`, 202, 206–207 (examples)
  real implementations, 239
Brown, Troy, *635*
bugs, μScheme, false reports of, 154
Byrd boxes, 139, 175

C programming
  advantages and disadvantages, 301
  conventions, 39–40
  interfaces, 40
  ML programming vs, 314–315
  scope rules, 74
  `static` variables, 124
`caar`, `cadr`, `cdar`, and so on, 96

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

calculational proofs, 114–116, *see also* equational reasoning
  form, 114
  organization, 115
calculus
  as a form of abstract machine, 241
`call/cc`, 172, 242–243, 244, 247, 255 (exercises)
call stacks, 65, 239, 244, 288
canvases (μSmalltalk), 618–619
capitalization
  in C programs, 39–40
  of value constructors, 469–471
capture
  of type variables, 369, 373
    in `type-lambda`, 365
capture-avoiding substitution, 165–167, 365, 373–375
  implementations, 374–375
  specification (rules), 374
  for type variables, 365
  `type-lambda` and, 377–378
`car`, 94, 172
  implementations, 162
  operational semantics, 151
  origin of name, 94
cascade, message (Smalltalk-80), 706
case expressions, 457, 459, 501
  algebraic laws, 476–477
  benefits, 501
  equational reasoning, 476–479
  evaluation, 490–494
  examples, 459–466, 475–476
  operational semantics, 490–494
  semantics, informal, 468–469
  type inference, 495–499
  typing rules, 349, 495–496 (nondeterministic), 497–499 (constraint-based)
`catch` (in μScheme+), 204
catching an exception, 208
categories, syntactic, *see* syntactic categories
`cdar`, `caar`, `cadr`, and so on, 96
`cdr`, 94, 172
  operational semantics, 151
  origin of name, 94
CESK machine, 243, 246
characteristic functions (of sets), 187
`check-assert`, 23, 26
`check-error`, 23, 25
`check-expect`, 23–26
  in μSmalltalk using =, 635
`check-principal-type`, 402
`check-type`, 402
checked run-time errors, 40
Church encodings
  Booleans, 655
Church-Rosser theorem, 246
`Circle` (μSmalltalk class), 620
circular lists, 673
  abstraction function, 673
  representation invariants, 673
class(es), 610, 708
  abstract, 622
  creation, 695
  definitions, 614
  modules vs, 588
  representations, 686, 694–695
`Class` (μSmalltalk class), 696–697
class `Collection`, 656–658
class definitions
  operational semantics, 685
class hierarchies
  collections, 642
  integers, 651
  numbers, 649
  Smalltalk-80, 704–707
class instance variables
  Smalltalk-80, 704
class `List`, 674–677
class `Magnitude`, 656
class `Metaclass`, 696–697
class methods, 614, 708
class `Object`, 696
class objects, 630
`Class` protocol, 638
class protocols, 613
class `True`, 655
class `UndefinedObject`, 696
class variables
  Smalltalk-80, 704
classifications
  of operators, 111–112, 546
clausal definitions, 318, 480 (defined), 501, 516
  benefits, 501
  examples, 480–482
clauses (Prolog), S95
client code, 526 (defined), 527, 530, 586
closing
  over free type variables, 414
closures, 122–125, 172
  application semantics, 147
  further reading, 175
  implementation, 122–124
  in μSmalltalk, 691
  in operational semantics, 147, 681
  representations
    C, 152
    nano-ML, 405
  semantics, 147–149
  static scoping and, 148
CLU, 526
  further reading, 588

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

CLU (continued)
mutable types vs immutable types, 545
program correctness and, 585
CNF, *see* conjunctive normal form
code chunks, 39
`coerce:` method, 663
coercions (μSmalltalk), 667–669
collection(s), 708
examples, 645
implementations, 656–662
inheritance, 649
μSmalltalk protocols, 642–649
`Collection` (μSmalltalk class), 656–658
collection hierarchies, 642
Smalltalk-80, 705–706
`Collection` protocol, 643, 644
coloring invariant, 261 (defined)
Common Lisp, 89, 174, 713
CommonLoops, 713
commutative diagrams, 477
compilation, 68
of pattern matching, 500–501, 503
compiling with continuations, 247
complex operations, 555–557, 662–673, 709
examples, 666
components of modules, 528 (defined)
declarations of, 538
exported, 538
of intersection types, 541
nested modules, 539
public, 538
strengthening, 570
types, 538
values, 538
composition
of functions, *see* function composition
of substitutions, 410–411
conclusions
in inference rules, 31
concrete syntax, 14, 15, 69
curried functions, impact on, 126
exceptions, 589
Impcore, 17–19
Impcore vs C, 12
μML, 467
μScheme, 92, 93
μSmalltalk, 627–629
rationale, 611
Molecule, 535
core layer, 536
module layer, 537
nano-ML, 404
partial application, impact on, 126
Smalltalk-80, 702–704
Typed Impcore, 330–331
concrete syntax (continued)
Typed μScheme, 352, 353
`cond` (Scheme form), 163–164
conditional expressions
algebraic laws, 112
implementations
μScheme+, 230–231
method dispatch vs, 623, 654–656
object-oriented programming, deprecated in, 655
operational semantics
Impcore, 34
μScheme, 149
small-step semantics, 217–218
type inference, 413, 418, 421
typing rules
nano-ML, 413 (nondeterministic), 418 (explicit substitutions), 421 (constraints)
Typed Impcore, 335
Typed μScheme, 363
conjunctions, 420
constraint solving, 429
conjunctive normal form, 139
representation (μScheme), 140
`cons`, 94, 172
implementations, 161, 473
operational semantics, 151
origin of name, 94
cons cell, 94, 95 (defined)
`Cons` protocol, 675
constraint(s), type-equality, 441
grammar, 420
implementations, 436
trivial, 420
in type inference, 420–428
typing judgment, 420
unsolvable, 424
constraint satisfaction, 428 (defined)
constraint solving, 418, 428–431, 441
conjunction, 429
examples, 431
implementations, 436–437
simple type equality, 429–430
constructed values, 457 (defined), 501
μML, 457
Molecule, 534
"constructor"
pitfalls as a technical term, 111
constructor(s), 111
algebraic data types and, 457
of Molecule values, 533
names for abstract syntax, 42
smart (in binary tries), 513
constructor functions
for records, 108
contexts, evaluation, *see* evaluation contexts

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

continuation(s), 136–138, 244
  blocks in Smalltalk as, 640
  compiling with, 247
  defined by evaluation context, 242
  delimited, 243, S358–359
  entering, 242
  exceptions and, 209–210
  failure, 136, 139, S239
  first-class, 242
  further reading, 246, 247
  in real languages, 242–243
  solutions, finding with, 142
  success, 136, 139
  undelimited, 243, 245
continuation-passing style, 138–143
  for backtracking, 138–143
  direct style vs, 138
  further reading, 175
  identifying, 138, 170
  in Smalltalk, 610, 640
  in Smalltalk conditionals, 655
`continue`, 202
  examples, 206–207
  real implementations, 239
contravariance
  of function arrow in subtyping, 606
`control` (control operator), 247
control flow, 201
  using continuations, 136–138
control operators, 201–202, 206–210, 244, 248–249 (exercises)
  big-step judgment forms, 677
  in big-step semantics, 679
  `break`, 202
  `continue`, 202
  `control`, 247
  `long-goto`, 204
  `long-label`, 204
  in μScheme+, 202–204
  μSmalltalk `return`, 677
  operational semantics, 210, 221–222
  `prompt`, 247
  in real languages, 239–240
  `reset`, 247
  `return`, 202
  `shift`, 247
  `throw`, 202
  `try-catch`, 204
conventions, coding
  C, 39–40
  ML, 301–303
`CoordPair`
  implementation, 615
  protocols, 613
copying garbage collection, 271–279
  C code, 276–278
copying garbage collection (continued)
  examples, 274–276
  in μScheme+, 276–278
  performance, 278–279
copying phase
  in μScheme+ garbage collector, 273
core languages, 26, 162, 214
  layer in Molecule, 534
  in μScheme, 120
Core μScheme+, 214
  operational semantics, 215–223
cost models of abstractions, 546
crash, μScheme, causes of, 154
creators (class of operations), 111
  introduction forms and, 347
  in protocols, 614
creators, producers, and observers, 111–112
  type systems and, 347
curried functions, 125–127
  concrete syntax and, 126
curried modules, 575
Curry-Howard isomorphism, 347, 727–729
currying and uncurrying, 125–127
  algebraic laws, 126
  higher-order functions, 126

Darcy, Lord, investigator for the Anglo-French empire, *S45*
data abstraction, 455, 525, 586, 709
  benefits, 525
  binary operations and, 662–673
  closed systems and, 626
  design, 545–554
  equality and, 634
  examples, 525–526, 546–548
  further reading, 587
  in object-oriented languages, 625–627
  open systems and, 626
`data` definitions
  μML, 466, 471–472
  typing rules, 487–489
database (of Prolog clauses), S46
datatype definitions, *see also* `data` definitions
  desiderata, 457
  generativity, 472
  μML, 471–473
  typing, 471–473
De Morgan's laws
  for `exists?` and `all?`, 131
  in solving Boolean formulas, 189
dead objects, 289
dead variables, 289

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

debugging
  closures (μScheme), S331
  garbage collectors, 280–283
  by tracing calls and returns, 198
  by tracing evaluation, 224
  by tracing message sends, 640, 641
decidability of type inference, 401
declarations
  of exported components, 538
  of manifest types, 530
  syntactic category of, 16
`define`
  desugared into `lambda`, 120
  explained, 22
  in full Scheme, 169
  operational semantics
    Impcore, 38
    μScheme, 152
  typing rules
    nano-ML, 416 (nondeterministic)
    Typed Impcore, 337
    Typed μScheme, 366
definition(s)
  class, 614
  extended, 18, 24–26
  Impcore, 27
  implementations of, 159–160
  syntactic category of, 16
  true, 18
definition modules, 528, 581
definitional interpreters, 69
degenerate type schemes, 414
delegation, 709
  in the implementation of `Set`, S563
delimited continuations, 243, 244, S358–359
  further reading, 247
denotational semantics, 246
dependent function types, 541
dependent types, 383, 727
derivations, 56
  construction (how to), 58–59
  validity, 57–58
desugaring, 68 (defined)
  `define`, 120, 152
  `do-while`, 66
  `let`, `let*`, `letrec`, 163
  `val-rec`, 396
  `while*`, 66
dictionaries, *see* finite maps
direct style, 138
  continuation-passing style vs, 138
dispatch, dynamic, *see* method dispatch
divergence, 245
`do:` (μSmalltalk method), 642–645, 656
  examples, 617, 622, 657
  `return` within, 657
domains
  of Molecule type-checking environments, 563
  of substitutions, 410
dot notation, 528
double dispatch, 666–667
duck typing, 627, 708
dynamic dispatch, *see* method dispatch
dynamic scoping, 148
dynamic typing, 327

eager evaluation, 241
EBNF, 17, S9–10
effects, 110
`either` (elimination form for sum types), 350
elaboration, 452 (exercise), 573 (defined)
  `exports`, 573
  generic-module types, 573
  intersection types, 573
  μML, 489
  module types, 564, 571–574
  module-arrow types, 573
  Molecule rules, 572
  operator overloading and, 577
  overloading (Molecule), 576
  types (Molecule), 564, 571–574
elimination forms, 347, 384, *see also* introduction and elimination forms
  exporting modules, 540
  generic modules, 541
  quantified types, 358
elimination rules, 345 (defined)
  template, 346
embeddings, 308, 318
  of functions, 312–314
  of lists, 308
  of ML primitives into μScheme primitives, 312
  projections and, 308–309
    of Booleans, 308
    of integers, 308
encapsulation, 586, 615
environments, 28–29, 69
  copying, 144
  global variables (Impcore), 28
  global variables (μSmalltalk), 678
  Impcore, 28
  implementation for Impcore, 54–55
  interface, 153
  lookup in Molecule, 574–576
  μScheme, 144
  representation in ML, 304–305
  typing
    Molecule, 562 (defined), 563
    nano-ML, 435

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

environments (continued)
typing (continued)
Typed μScheme, 361
equality
algebraic laws, 104
of association lists, 132–133
in C, 633
of collection elements, 646
data abstraction and, 634
exercises, 718
in full Scheme, 633
in μScheme, 633
in OCaml, 634
overloading, 543
polymorphic, 633
of S-expressions, 104
in Standard ML, 633
syntactic form for (Typed Impcore), 332
of types (pitfalls), 332
typing rules (for Typed Impcore), 336
equality constraints, *see* type-equality constraints
equality types (Standard ML concept), 633, S217
equational reasoning, 113–116, 183
with case expressions, 476–479
equivalence
identity vs (in collections), 646
μSmalltalk, 635
observational, 634
of types, *see* type equivalence
erasure (of types), 380
Erlang pattern matching, 499
error(s)
categories of, 13–14
run-time, 40, 47
syntax, 47
`:error` exception, 208–209
evaluation, 69
`block`, 691
case expressions, 468–469
eager, 241
Impcore definitions, 22–23
Impcore expressions, 20–22
lazy, 241
message sends, 689–690
μSmalltalk literals, 691
μSmalltalk primitives, 692–693
modules, 541–542
Molecule, 537
overloading (Molecule), 577
polymorphism and, 380
`return` (μScheme+), 237
`return` (μSmalltalk), 689
stack-based, 212–213, 227–239
strict, 241
evaluation (continued)
tracing, 224–225
Typed μScheme, 380
evaluation contexts, 202, 241, 245
stack frames and, 242
evaluation order, 242
evaluation stacks, 210–213, 245, 289, 730
analogs in real languages, 239
exercises, 250
instrumentation, 224–225
interface to (μScheme+), 224
memory management and, 229–230
evaluators, 69
case expressions, 492–494
Impcore, 48–54
μScheme (C), 154–160
μScheme (ML), 309–312
μSmalltalk, 688–694
exceptions, 207–208, 318, 583
abstract types and, 583–584
catching, 208
continuations and, 209–210
examples, 208–209
implementations, 208, 246
raising, 207, 583
in real languages, 240, 583
syntax, 589
throwing, 207
execution, *see* evaluation
exhaustive pattern matches, 318
checks for, 499
existentially quantified types, 503
from algebraic data types, S25
export lists, 538
exported components, 538
exported names, 526, 527
exporting modules, 539 (defined)
`exports` module types
elaboration, 573
expression(s)
syntactic category of, 16
type-level, 352, 357
expression-oriented languages, 17, 69
Extended Backus-Naur Form, 17, S9–10
extended definitions, 18, 24–26

facts (in Prolog), S45, S96
failure continuations, 136, 139
in `withHandlers`, S239
filtering, 172
examples, 206
higher-order functions, 127
methods, 645
visualizations, 129
`find`
algebraic laws, 106
using continuations, 136–137

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

finite maps, 105
  algebraic laws, 112
first-class functions, *see* functions, first class
first-class values, 121
fish in a barrel, shooting, *S448*
flips (copying collection), 271
`Float` protocol, 652
folding, 173
  higher-order functions, 128
  to implement set functions, 131
  methods, 645
  visualizations, 129
F_ω, 365
form(s)
  judgment, 30
  syntactic, 15–17
formation
  of generic-module types, 541
  of intersection types (Molecule), 541
  of module types, 538–539
formation rules, 345 (defined), 347, 384, *see also* type-formation rules
  template, 346
formation, introduction, and elimination, 347
  Molecule, 538
forwarding (action), 277–278
forwarding pointers, 272–273, 289
`Fraction` protocol, 652
fraction(s) in μSmalltalk
  examples, 652
  implementation, 664–666
frames (on evaluation stacks), 210–211
free lists, 266, 289
free occurrences
  of type variables, 371
free type variables, 371 (defined)
  implementations, 371, 372
  inference rules, 371
free variables, 173, 315–317, 318
  formal definition, 316–317
  in μScheme expressions, S328
fresh locations, 144, 145
  in function application (μScheme), 147
fresh type constructors, 484
fresh type variables, 433–434
fresh variables, 165, 195, 196, S217
freshness and generativity, 484
from-space, 271 (defined)
ftv (compute free type variables), 414
  in nano-ML, 433
function(s)
  anonymous, 120
function(s) (continued)
  first-class, 120–122, 172
    first-class, nested functions vs, 121
    implementation, 122–124
  first-order, 90, 122, 172
  higher-order, *see* higher-order functions
  nested, *see* nested functions
  primitive
    environment ϕ and, 30
  typing rules, 348
  user-defined
    in Impcore environment ϕ, 30
  variadic, 169, 322 (exercise)
`function?`, 92
function application
  Impcore, 36
  implementations
    Impcore, 50
    μScheme (C), 156–157
    μScheme (ML), 310
    μScheme+, 231–234
  Lisp, 148
  μScheme, 147
  operational semantics
    Impcore, 36, 37
    Lisp, 148
    μScheme, 147, 150
  small-step semantics, 218–220
  type inference, 419, 421
  typing rules
    explicit substitutions, 419
    nano-ML, 414 (nondeterministic), 421 (constraint-based)
    Typed Impcore, 336
    Typed μScheme, 365
function composition, 125
function environment ϕ (Impcore), 30
function values, 120–122
functional abstraction, 36
functional programming
  object-oriented programming vs, 624–625
  procedural programming vs, 205–206
functors
  generic modules, 581
  ML, 581
  Prolog, S43, S52

GADTs, 503, S25, S32–37
garbage collection, 259 (defined), 289
  benefits, 257
  bugs, common, 280
  compacting, 288
  concurrent, 286
  conservative, 283, 288

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

garbage collection (continued)
copying, 271–279, *see also* copying garbage collection
debugging, 280–283, 291
further reading, 291–292
generational, 286, 289
manual memory management vs, 291
mark-and-sweep, 266–271
mark-compact, 283
overhead, 259
parallel, 286
performance, 260, 270
in real systems, 285–287
write barrier, 286
general, at least as, 410, 441
generality
of types, 410
generalization, 413, 423–424, 442
implementations, 434
`let` binding, 425–428
rationale, 435
soundness, 427
of types, 415 (defined), 423
generalized algebraic data types, *see* GADTs
generational garbage collection, 286
generativity, 383, 483–485 (defined), 501
in C, 484
datatype definitions, 472
in μML, 484
in Molecule, 559, 578
in real languages, 580
of sealing in Molecule, 539
specification (μML), 487
in Standard ML, 484
of syntactic forms, 484
generic(s), *see* polymorphism
generic (defined by recursion over types), 132
generic (parametrically polymorphic), 132, *see also* polymorphism
generic module(s), 526, 540–541, 586
curried, 575
elimination of, 541
examples, 552, 560
instantiation of, 541
introduction of, 541
in real languages, 580
generic-module types
elaboration, 573
examples, 540
generic programming, 132
global variables
environments (Impcore), 28
environments (μSmalltalk), 678
semantics, 151–152
goals (Prolog), S49, S96
going wrong, 22, 328
`goto`, 201
considered harmful, 201, 245
grammars, 15, 69
gray objects, 261
in copying collection, 273
examples, 275
in mark-sweep collection, 269
ground clauses (Prolog), S53
ground terms (Prolog), S53, S96
ground types, 408

`has?` (μScheme function)
algebraic laws, 103
Haskell, 441
head(s)
of Prolog clauses, S52
of Prolog rules, S97
headroom
in garbage collection, 259, 270
heap(s), 289
heap allocation, 289
in μScheme+, 263
heap growth
in copying collector, 278
heap invariants
array heaps, 552
leftist heaps, 556
heap(s), leftist, 555–557
heap objects, 289
heap pointer, 271 (defined)
`heaplimit`, 271
Heidi (broadcast on NBC), *526*
higher-order functions, 90, 121–122, 173
algebraic laws (lists), 130
for control flow, 136–138
currying and uncurrying, 126
for embedding, 312–314
exercises, 187–189
implementations, 129–131
on lists, 127–131
filtering, 127
searching, 127–128
transformation (mapping), 127
for random-number generation, 124–125
Hindley-Milner type system, 401
in real languages, 441
Hoare triples (as metaphor for moves in blocks world), S83
holes
in Molecule typing contexts, 562
in small-step semantics, 210–222, 245
in stack frames, 210–223, 226
Horn clauses, S46, S91
`hp` (heap pointer), 271

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

hygiene, 166, 171
  in full Scheme macros, 171
  further reading, 175
  in substitution, 165–167
  in translations for μScheme, 162

idempotence (of substitutions), 411
identifier cross-reference in this book, 39
identity
  equivalence vs (in collections), 646
identity substitution, 411
`idsubst` (identity substitution), 411
`if` expressions, *see* conditional expressions
`IFX` vs `IF`, 42
ill-behaved programs, 14
ill-formed programs, 13
ill-typed programs, 13
images, Smalltalk, 701
immutability
  of abstractions, 545
Impcore, 12–87
  abstract syntax, 27–28
  concrete syntax, 12, 17–19
  environments, 28, 54–55
  evaluation, 20–23
  initial basis, 27
  interpreter
    environments, 44–45
    `eval`, 48–54
    interfaces, 41–47
    values, 43
  local variables, 66, 86
  operational semantics, 32–38
  semantics, informal, 20–22
  Simplified, 62
imperative features, 402, 403
implementation(s)
  allocation, 162 (μScheme)
implementation modules, 527, 581
`implicit-data`, 472–473
impredicative polymorphism, 540, 577
  in modules, 578
impure languages, 403
induction principles, 116
inductively defined data, 116
  specified by recursion equations, 117
inference rules, 31–32, *see also* operational semantics, typing rules
  binary trees, defining, 109
  case expressions
    evaluation, 490–494
    type inference, 495–499
  `data` definitions (μML), 487–489
inference rules (continued)
  elaboration
    μML, 489
    Molecule, 572
  evaluation
    case expressions, 490–494
    Impcore, 32–38
    μScheme, 145–152
    μScheme+, 216–223
    pattern matching, 490–494
  form of, 31
  free type variables, 371
  free variables, 316–317
  instantiation, 56
  lists, defining, 116
  overloading (Molecule), 576
  pattern matching
    evaluation, 490–494
    type inference, 495–496
  as Prolog code, S45
  S-expressions, defining, 117
  soundness, 419
  substitution, capture-avoiding, 374
  type compatibility, 487–489
  type equivalence, 369–370
  type translation (μML), 489
  types, *see* typing rules
information hiding, 455, 586
  further reading, 587
inhabitants
  of intersection types, 541
  of a subtype or supertype, 561
  of types, 334, 388
  of the unit type, 349
inheritance, 610, 709
  collections and, 649
  effect on dispatch, 631–633
  examples, 620–622
  of instance variables, 614
initial basis, 26 (defined), 69
  Impcore, 30
  μScheme, 97
  μSmalltalk, 636–653
  Molecule, 544
  nano-ML, 440
  Typed μScheme, 382
initialization
  local variables (μScheme), 117
  μSmalltalk examples, 615–616, 621
  of Smalltalk objects, 616, 624
insertion sort, 101
  algebraic laws, 101
instance(s), 442, 709
  of classes, 610
  of generic modules, 541
  of inference rules, 56
  of polymorphic functions, 356
  as qualified names, 541

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

instance(s) (continued)
of templates, 166
instance methods, 614
instance protocols, 613
instance relation between types ($\leqslant$), 410
instance variables, 614, 624, 709
inheritance of, 614
in the μSmalltalk interpreter, 686
visibility of, 624
instantiation, 384, 409–413, 442
automatic, 409
examples (generic modules), 560
examples (types), 358–359
explicit, 409
of generic modules, 541, 560
typing rule, 575
implementations, 376, 411, 434
implicit, 409
of inference rules, 56
mistakes, common, with, 358
of ML types, 304
of ML type schemes, 410
of polymorphic functions, 356
of polymorphic values, 304 (ML), 371–376
quantified types and, 358
typing rule, 365
rationale, 435
of templates for syntactic sugar, 166
typing rules
generic modules, 575
quantified types, 365
integer(s), *see also* natural numbers
implementations (μSmalltalk), 664, 666–667
large, 666
small, 666
integer hierarchy, 651
`Integer` protocol, 651
interfaces
in C code, 40
design, 545–554
design examples, 546–548
memory management and, 257
Molecule, 528–532
in real languages, 580–583
separate compilation and, 580
Smalltalk, *see* protocols
`InternalError` exception, S219
interpreters
Impcore, 38–55
μScheme (C), 152–162
μScheme (ML), 303–314
μScheme+, 223–239
μSmalltalk, 685–700
Molecule, 579, S475–524
nano-ML, 433–440
interpreters (continued)
Typed Impcore, 337–343
intersection types
"and" operator ($\wedge$) and, 564
components, 541
elaboration, 573
formation, 541
idioms, 541
inhabitants, 541
Molecule, 541
principal module types and, 567
subtyping of, 564
introduction and elimination forms
creators, producers, and observers related to, 347
introduction forms, 347, 384
for functions, 348
for generic modules, 541
for modules, 539
for pairs, 348
for products, 348
for quantified types, 359
for sums, 349
introduction rules, 345 (defined)
template, 346
invariants, 586, 673–677, *see also* representation invariants
of stack-based `eval`, 227
`isKindOf:`
correct usage, 637
`isMemberOf:`
correct usage, 637
`it` (automatically bound variable)
in Impcore, 54
in μScheme, 160
iterators
observers, as a species of, 645
in Smalltalk, 645

Java
abstract types, 707
Smalltalk vs, 707
judgment(s), 30–31, 69
judgment forms, xviii (table), 69
expressing "may differ", 33
expressing "must differ", 33
expressing "must equal", 33

keyed collections
implementations, 659–660
μSmalltalk protocols, 646, 659
`KeyedCollection` protocol, 647
keys (in association lists), 105
kind(s), 351–355, 384
checking, 378–380
implementation, 355–356
μML, 466
in nano-ML, unneeded, 409

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

kind(s) (continued)
of primitive type constructors
Typed μScheme, 356
slogan, 357
of type constructors, 352
kinding judgments, 354, 355

`lambda`, 120–125
dynamic scoping, 148
exercises, 187–189
implementations (C), 156
implementations (ML), 310
operational semantics, 147
small-step semantics, 217
typing rules
nano-ML, 414 (nondeterministic), 446 (constraint-based)
Typed μScheme, 365
lambda abstraction, 120–122, 173
lambda-bound variables, 414, 442
polymorphism and, 415
lambda calculus, 241
inspiration for Lisp, 89
lambda expression, *see* `lambda`, lambda abstraction
Lambda: The Ultimate (further reading), 174
`LAMBDAX` vs `LAMBDA`, 42
large integers, 666, *see also* multiprecision arithmetic, natural numbers
invariants, 557
`LargeInteger`
class definition, 669
private methods, 668
protocol, 651
laws, algebraic, *see* algebraic laws
lazy evaluation, 241
lazy sweeping, 266
leftist heaps, 555–557
length of a list
algebraic laws, 98
proof of laws, 114
`let`
implementations
μScheme (C), 157–158
μScheme (ML), 310
μScheme+, 231–232, 234–236
Milner's, 414, 425–428
operational semantics, 146
small-step semantics, 220
syntactic sugar, 163
typing rules
nano-ML, 415 (nondeterministic), 426 (constraint-based)
Typed μScheme, 363
let binding, 117–120
let-bound variables, 414, 442

`letrec`, 120, 135
implementations
μScheme (C), 158–159
μScheme (ML), 311
μScheme+, 234–236
operational semantics, 146–147, 364
polymorphism and, 416
small-step semantics, 221
syntactic sugar, 163
type annotations, 364
type inference, 415, 420
typing rules
nano-ML, 415 (nondeterministic), 420 (substitutions)
Typed μScheme, 364
`let*`
assignment vs, 119
implementations
μScheme (C), 158
μScheme (ML), 310
μScheme+ (lowering), 214, S350
in μScheme, 118
operational semantics, 146
syntactic sugar, 163
typing rules
Typed μScheme, 364
`LETX` vs `LET`, 42
lexical analysis, S191
lexical scoping, 148
lexical structure, 14, 15
of all bridge languages, 17
μML value constructors, 469–471
lies, damn lies, and the logical interpretation of Prolog, *S63*
linear search
higher-order functions, 127–128
Lisp
bugs, 148
conditional expressions, *see* `cond`
function application, 148
inspired by lambda calculus, 89
recursion in, 90, 98
list(s)
algebraic laws, 130
`append` laws, 99
in C interpreters, 45–46
circular, 673
constructors, 318
filtering
function `filter`, 127
methods `select:` and `reject:`, 645
visualizations, 129

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

list(s) (continued)
folding
functions `foldl` and `foldr`, 128
method `inject:into:`, 645
visualizations, 129
higher-order functions, 127–131
inference rules, defined by, 116
insertion laws, 101
mapping
function `map`, 127
method `collect:`, 645
visualizations, 129
μML, 473
μScheme, 94
μSmalltalk examples, 648
μSmalltalk methods, 643–645, 649
operations, 98–101
notation, condensed, 99
pattern matching on, 475–476
primitives' laws, 111
principles for programming with, 98–107
representation for sets, 104–105
reversal, 100
searching
functions `exists?` and `all?`, 127–128
method `detect:`, 643
selection
function `filter`, 127
methods `select:` and `reject:`, 645
visualizations, 129
transformation
function `map`, 127
method `collect:`, 645
`list`
full Scheme function, 96
μScheme primitive, 198 (exercise)
`List` (μSmalltalk class), 674–677
list comprehensions
in a semantics for Prolog, S110
how to implement, S114
`List` protocol, 649
literal expressions
in μScheme, 95
μSmalltalk
bootstrapping, 697–699
evaluation, 691
operational semantics
μSmalltalk, 680
small-step semantics, 216
Smalltalk-80, 701
typing rules
Typed Impcore, 335
Typed μScheme, 363
literate programming, 39
live data, 289
local variables
initialization (μScheme), 117
μScheme, 117–120
location(s), 173
fresh, 144, 145
of μScheme+ objects, 258
Molecule, 535
mutable, 122–124, 144, 145, 535
location semantics, 144–152, 173, 677–685
logic (as a programming language), S43–47
logic programming, S96
logical variables, S45, S96
`long-goto`, 204, 208
exercises, 251
implementations, 237
small-step semantics, 221–222
`long-label`, 204, 208
implementations, 236–237
small-step semantics, 221–222
loop(s)
operational semantics
Impcore, 35
μScheme, 149
typing rules
Typed Impcore, 336
Typed μScheme, 363
loop invariants, 586
lowering
implementation, 226
rules (μScheme+), 214, 216
semantics, 213–215
Lua, for object-oriented programming, 712

macros, 171, 373
further reading, 175
hygienic, 171 (defined)
magnitude(s), 709
implementations, 656
`Magnitude` (μSmalltalk class), 656
`Magnitude` protocol, 649–651
major (garbage) collection, 286
managed heaps, 259 (defined), 289
allocation in μScheme+, 263
size and growth, 262–263
manifest types, 530, 538
in C, 39
as components, 538
in real languages, 580
mapping, 173
higher-order functions, 127
methods, 645
visualizations, 129
mark bits, 266, 290

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

mark phase
in μScheme+ garbage collector, 268–270
mark-and-sweep garbage collection, 266–271, 290
performance, 270–271
mark-compact garbage collection, 283
marker methods, 636
match compilation, 500–501, 503
matching, *see* pattern matching
materialization, 179
"may differ," in judgment forms, 33
`member?` (μScheme function), 187
membership in S-expressions
algebraic laws, 103
memory management, 257–299
evaluation stacks, 229–230
explicit, 289
reference counting, 283–285
memory safety, 240, 257, 290
in C programs, 385
message(s), 709
private, *see* private messages
to `super`, 621
message cascades (Smalltalk-80), 706
message categories, 613
in Smalltalk-80, 703
message names, 612
arity, 627
message not understood, 631, 709
message passing, 609, 709
basics, 610–611
message patterns
in Smalltalk-80, 703
message selectors, 612 (defined), 631, 709
message sends
evaluation, 689–690
operational semantics, 681–682
metaclass(es), 630 (defined), 694–695
in the μSmalltalk interpreter, 686
representation, 686
`Metaclass` (μSmalltalk class), 696–697
metalanguages, 308 (defined), 318
metatheoretic proofs, 59–66, 69, 116
about data, 116–117
construction (how to), 61–65
example, 63–65
small-step semantics and, 244
utility of, 65–66
metatheory, 59–66, 69, *see also* metatheoretic proofs
exercises, 195
metavariables, 70
in operational semantics, 31
program variables vs, 19
for syntax, 19
method(s), 609, 614, 709
class, 614
complex, 662–673
private, *see* private methods
representations, 677, 686
method dispatch, 610, 613, 631–633, 710
conditionals vs, 623, 654–656
examples, 632
implementation, 690
in Smalltalk, 623
μML, 457–499
abstract syntax, 485–486
concrete syntax, 467
predefined functions, 475–476
predefined types, 464, 473–475
values, 468, 486
μScheme, 90–172
abstract syntax, 144–145
concrete syntax, 93
initial basis, 97
predefined functions, 96–100, 104–107, 125–128, S319–320
primitive functions, 92–96
primitives, 150–151
values, 91–92, 145
μScheme+, 202–239
abstract syntax, 225–226
introduction, 202–205
μSmalltalk, 610–700
abstract machine, 678–679
behaviors, 678–679
concrete syntax, 627–629
rationale, 611
initialization, 616
interfaces, *see* protocols
predefined classes, 636–653
primitive classes, 696–697
primitives, 699–700
operational semantics, 682
values, 629–630
Milner's `let`, 414, 425–428
Milner, Robin, 320, 401
minor (garbage) collection, 286
mixed arithmetic
in μSmalltalk, 651, 723–724
in Smalltalk-80, 705
ML programming
advantages and disadvantages, 301–303
C programming vs, 314–315
conventions, 301–303
quick reference, 320
Modula-2, 586, 588
Modula family, 526
modular type checking, 558
further reading, 588
in real languages, 580

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

module(s), 525, 527, 587
as components, 539
definitions, 539
desiderata, 527, 581–583
elimination forms, 540
evaluation, 541–542
examples, 528–529
exporting, 539
further reading, 588
generic, *see* generic modules
introduction forms, 539
nested, 539
in real languages, 581
objects and classes vs, 588
parameters, 540
polymorphism, 540–541
programming style, 526
in real languages, 527, 580–583
recursive, 588
representations, 541–542
uses, 540
module arrow `--m->`, 540
module-arrow types
elaboration, 573
module types, 528, 587
elaboration, 571–574
formation, 538–539
generic modules, 541
independence from modules, 529
principal, *see* principal module types
in real languages, 581
realization, 568
rooted, 563
strengthening, 559, 570
subtyping, 564–566
uninhabited, 566
Molecule, 526–580
concrete syntax, 536, 537
core layer, 534–538
module layer, 538–542
overloading, 543–544, 576–577
predefined modules, 544
primitive types, 544
Smalltalk vs, 615
subtyping, 564–566
syntactic categories, 539
type system, 558–579
design goals of, 558
overloading, 576
overview, 562
values, 535
monomorphic functions, 131, 173
monomorphic type systems, 331, 344, 384
limitations, 351
monotypes, 384, 408, 442
`msubsn`, 567 (defined)
multiple arguments
functions consuming, 104
multiple inheritance, 710
multiple representations
with data abstraction, 555–557
in object-oriented programming, 662–673
multiprecision arithmetic, 557
invariants, 557
μML, alphabetized as micro-ML
μScheme, alphabetized as micro-Scheme
μSmalltalk, alphabetized as micro-Smalltalk
"must differ"
in judgment forms, 33
"must equal"
as constraint $\sim$, 420
in judgment forms, 33
mutability, 173, 545
of abstractions, 625
allocation and, 545
Molecule, 534, 535
of queues, S146
in Smalltalk, 625
mutable locations, *see* locations, mutable
mutable reference cells, 188, 318
mutable state
of a machine, 11
in μSmalltalk class representations, 694
shared, 174
in a resettable counter, 124
mutable variables
avoided by `let*`, 119
mutablilty
of abstractions, 545
mutation, 402
in full Scheme, 169
observational equivalence and, 636
mutators (class of operations), 112, 259, 290
in protocols, 614

name(s)
μSmalltalk denotations, 631
qualified, 528, 540
of value constructors for abstract syntax, 42
name equivalence, 383, *see also* generativity, type equivalence
name resolution
dynamic, 630
μSmalltalk, 630–631
static, 630

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

nano-ML, 401–440
  abstract syntax, 404–405
  concrete syntax, 404
  μScheme vs, 401
  operational semantics, 405–407
  primitive type constructors, 412
  type system, 407–417
`Natural`
  class-definition template, 670
  private methods, 671, 672
  protocol, 652–653
natural deduction, 32, 70
natural numbers, S13–22
  abstraction functions, representations, and invariants, 669–673, S14–15
nested functions, 121–122, 173
nested modules, 539
  in real languages, 581
nested patterns, 460–461
`new` (μSmalltalk message), 624
  examples, 621
`new` method
  implementation, 695
Newton-Raphson technique, 652
`nil` (μSmalltalk object), 630
  implementation, 696
  other languages vs, 630
nondeterminism
  in calculi, 241
  in rules for nano-ML, 413–416
  in semantics, 241
  in typing, 416–417
nonterminal symbols, 16
`:not-found` exception, 209–210
Noweb, 39
`null?`, 92, 94
nullary type constructors, 343
number(s), 710
  implementations (μSmalltalk), 663–664
  Smalltalk-80, 705
`number?`, 92
number hierarchy, 649
`Number` protocol, 649–653
numeric coercions
  μSmalltalk, 667–669

O Lochlainn, Sean, forensic sorcerer, *S45*
object(s), 455, 525, 609, 710
  abstract types vs, 625–627
  basics, 610–611
  equality and, 634
  modules vs, 588
  in Prolog, S43
object(s) (Prolog), S96
`Object` (μSmalltalk class), 696
  implementation, 696
  protocol, 637
object identity, 634
  observational equivalence vs, 635
object languages, 308, 319
object-oriented interfaces, *see* protocols
object-oriented languages
  origins and evolution, 609
object-oriented programming, 455, 587
  abstract classes, 656–662
  binary operations, 662–673
  classes, without, 707
  decision-making code, 654–656
  functional and procedural programming vs, 624–625
  historical development, 712
  invariants and, 673–677
  in Lua, 712
  sums of products in, 466
  technique, 654–677
Objective C, 712
observational equivalence, 110, 634
  in Impcore exercise, 77
  μSmalltalk, 635
  mutation and, 636
  object identity vs, 635
observers (class of operations), 111, *see also* creators, producers, and observers
  elimination forms and, 347
  in protocols, 614
OCaml, 399
  equality, 634
occurrence equivalence, 383, *see also* generativity, type equivalence
occurs check, S96
  in Prolog, S60
  in type inference, 429
operational semantics, 29–32, 68, 70, *see also* small-step semantics
  abstract-machine, 211
  assignment (`set`)
    Impcore, 34
    μScheme, 145
    μSmalltalk, 680
  big-step, 31
  blocks, 681
  `car`, 151
  `cdr`, 151
  class definitions, 685
  closures, 147, 681
  conditional expressions (`if`)
    Impcore, 34
    μScheme, 149
  `cons`, 151

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

operational semantics (continued)
control operators, 210, 221–222, 683–684
Core μScheme+, 215–223
`define`
Impcore, 38
μScheme, 152
dynamic scoping, 148
function application
Impcore, 36, 37
Lisp, 148
μScheme, 147, 150
global variables, 151–152
Impcore, 32–38
`lambda`, 147
`lambda` (dynamic scoping), 148
`letrec`, 146, 146, 364
literal expressions
μSmalltalk, 680
loops
Impcore, 35
μScheme, 149
lowering, 213–215
message send, 681–682
μScheme, 144–152
μSmalltalk vs, 677–678
μScheme+, 210–223
μSmalltalk, 677–685
μScheme vs, 677–678
μSmalltalk primitives, 682
mutation-free, 405–407
nano-ML, 405–407
natural-deduction, 32
nondeterministic, 241
reduction, 215
`return`
μScheme+, 222
μSmalltalk, 683–684
sequencing (`begin`)
Impcore, 35
μScheme, 149
μSmalltalk, 679
small-step, *see* small-step semantics
stack-based, 210–213
`super`, 680
uses, 240–242
`val`
Impcore, 38
μScheme, 151
μSmalltalk, 685
nano-ML, 407
variables
Impcore, 32
μScheme, 145
μSmalltalk, 680
nano-ML, 405
operator classifications, 111–112, 546, *see also* creators, producers, and observers
optimized tail calls, 170, 214
exercises, 251–252
further reading, 246
implementations, 238–239
in real languages, 239
order of evaluation, 35, 403
`order` type (μML), 474
overheads
copying garbage collection, 279
mark-sweep garbage collection, 270
overloading, 132, 383, 543–544
elaboration (Molecule), 576
evaluation (Molecule), 577
examples, 543–544
Molecule, 543, 576–577
parametric polymorphism vs, 383
in real languages, 584–585
resolution, 544
via type classes, 585, 589
overriding (of methods), 614, 632, 710

$\mathcal{P}$ (primitive function in identifier cross-reference), 39
pages (units of heap), 267–268
pair type(s)
algebraic laws, 349
typing rules, 348–349
`pair` type (μML), 464
pangrams, 103
parameters, accumulating, 101
parametric polymorphism, *see* polymorphism, parametric
partial application, 125–127
paths, *see* access paths
pattern(s), 319, 501
exhaustive, 499
μML, 466
nested, 460–461
overlapping, 500
redundant, 319, 500
pattern matching, 319, 458, 501
benefits, 501
compilation, 500–501, 503
compile-time checks, 499–500
efficiency, 500–501, 503
equational reasoning, 476–479
in Erlang, 499
evaluation, 490–494
examples, 307, 309–312, 459–466, 475–476
exhaustive, 499
in ML, 307
nested patterns, 460–461
operational semantics, 490–494

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

pattern matching (continued)
semantics, informal, 468–469
syntactic sugar, 480–483
type inference, 495–499
typing rules, 495–496 (nondeterministic), 497–499 (constraint-based)
ubiquitous, 480–483
wildcards, 461
phantom types, S37
Pharo (Smalltalk system), 712
phase(s)
in language processing, 13–14
phase distinction, S392
π, *see* access paths
π-calculus, 241, 247
pictures (μSmalltalk example), 616–618
PLT Redex, 246
polymorphic equality, 633
polymorphic functions, 131
recursive, 359–360
polymorphic type(s), 319
of list functions, 358
polymorphic type systems, 328, 351–382, 384
advantages, 351
origins, 385
polymorphic, recursive functions
in Typed μScheme, 359
polymorphism, 132, 173, 332, 344, 384
ad hoc, 132, 544, *see also* overloading
impredicative, 540, 577
in modules, 578
lambda-bound variables and, 415
in μScheme, 131–136
in ML, 304
modules, 526
in Molecule
nano-ML vs, 577
Typed μScheme vs, 577–578
in nano-ML vs Molecule, 577
parametric, 132, 384, 401
of lists, 116
predicative, 540, 577
programming technique, 133–135
in Scheme, 133–135
species of, three, 132
subtype, 132, 646
in Typed μScheme vs Molecule, 577–578
polytypes, 385, 408, 442
limited to type environments, 413
predefined functions, 26, 70, *see also* initial basis
Impcore, 27
μML, 475–476
predefined functions (continued)
μScheme, 96–100, 104–107, 125–128, S319–320
nano-ML, 440
Typed Impcore, 331
predefined modules
Molecule, 544
predefined types
μML, 464, 473–475
predicate(s) (in Prolog), S45, S96
predicate logic, S96
predicate transformers, 246
predicative polymorphism, 540, 577
premises
in inference rules, 31
of a Prolog clause, S52
preservation (proof technique), 240
primitive classes (μSmalltalk), 696–697
primitive functions, 26, 70, *see also* initial basis
environment ϕ and, 30
Impcore, 23–24
μScheme, 92–96, 150–151, 160–162, 306, 312–314
nano-ML, 440
Typed μScheme, 381–382
primitive type constructors
Molecule, 544
nano-ML, 412
Typed μScheme, 381
primitives, μSmalltalk, 699–700
evaluation, 692–693
operational semantics, 682
principal module types, 561, 566–571
computing (how to), 568–571
intersection and, 567
subtyping and, 566
principal type(s), 416–417, 442
testing, 402, 417
principal type schemes, 417
`printu`, 23
private messages
in μSmalltalk collection classes, 658
`Natural` (μSmalltalk class), recommended for, 670, 672
in Smalltalk, 623
private methods, 616, 623, 710
examples, 668
μSmalltalk integer classes, 668
`Natural`, 671, 672
representations, exposing with, 664
in Smalltalk, 710
procedural programming, 11, 70, 201–202, 204
functional programming vs, 206–207

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

procedural programming (continued)
object-oriented programming vs, 624–625
producers (class of operations), 111, *see also* creators, producers, and observers
introduction forms and, 347
in protocols, 614
product types, 465
algebraic laws, 349
typing rules, 348
program variables, *see also* variables
metavariables vs, 19
program verification, 553
programming conventions
C code, 39–40
ML code, 301–303
progress and preservation (proof technique), 240
projections, 308, 319, *see also* embeddings
Prolog, S43, S95
arity, S50
"backward" execution in, S56
difference lists, S76
logical interpretation, S55
occurs check, S96
primitive predicates, S68
procedural interpretation, S61
logical interpretation vs, S63, S95
real, S90
semantics, S91
syntax, S90
semantics, S55
syntax, S52
unification, S60
`prompt` (control operator), 247
proof(s)
of abstract types, 588
calculational, *see* calculational proofs, equational reasoning
derivations, about, 59–66
derivations as, 56–59
metatheoretic, 59–66
about data, 116–117
in operational semantics, 55–66
by structural induction, 115
proof principles
induction, 116
proof techniques
small-step semantics, 240
proper tail recursion, 170, 174, *see also* optimized tail calls
further reading, 246
propositional logic, S97
propositions as types, 347, 727–729
protocols
of objects, 613
prototypes
in object-oriented programming languages, 707
public components, 538
public names, 527
pure expressions, 110
pure languages, 403

qualified names, 528, 540
from generic modules, 541
quantification, existential
in algebraic data types, S25
quantified types, 351, 356–360, 385
elimination form, 358
introduction form, 359
in nano-ML, 407–408
quasiquotation, 169, 197 (exercise)

Racket (Scheme dialect), 168
raise an exception, 583
random-number generation
higher-order functions, 124–125
rational numbers
in μSmalltalk, 664–666
reachability, 258–261, 290
examples, 258–259
in μScheme+, 263–264
realization
of module types, 567, 568
receiver(s), 610, 612, 710
receiver-first syntax
rationale, 611
records
encoded as S-expressions, 107–109
exercises, 181, 196
μScheme, 107–109
Molecule, emulation in, 533
syntactic sugar, 167–168
Molecule, 542
types, 383
recursion
difference lists, avoiding with, S77
example of evaluation, 99
on lists, 98–107
modules, 588
open, 710
recursion equations (to define data), 117
recursive definitions, 120
recursive evaluation
stack-based evaluation vs, 211–213
recursive functions
`letrec` and, 120, 135
polymorphic, 359–360
recursive, polymorphic functions
in Typed μScheme, 359

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

recursive types, 465
examples, 463–464
redefinition
of Smalltalk methods, 614
Redex, 246
reduction
in a calculus, 241
in parsing, S194
reduction semantics, 241, 245
further reading, 246
redundant patterns, 319, 500
reference cells, mutable, 188, 318
reference counting (memory management), 259, 283–285, 290
reflection, 711
in μSmalltalk, 630
in Prolog, S93
in Smalltalk-80, 630, 706–707
relations, S97
as a programming tool, S43–47
rely-guarantee reasoning, 587
to satisfy invariants, 552
remembered set, 286
renaming
implementations, 375
of type variables, 368–369
of variables, S58
representation invariants, 545, 550, 587, 711
binary search trees, 550
circular lists, 673
complex numbers, 551
data structures, typical and, 549–551
dictionaries, 549
environments, 54
examples, 551–554, 556, 664
floating-point numbers (μSmalltalk), 652
fractions (μSmalltalk), 664
large integers, 557
leftist heaps, 556
metaclasses, 694
natural numbers, 670, 672
priority queues, 550, 552
rational numbers, 664
sets, 548
two-dimensional points, 551
reserved words, 19
`reset` (control operator), 247
`return`, 202
big-step judgment forms, 677
in `do:` loop (Smalltalk), 657
evaluation
μScheme+, 237
μSmalltalk, 689
μScheme+ control operator, 202
μSmalltalk control operator, 677
`return` (continued)
operational semantics
μScheme+, 222
μSmalltalk, 683–684
real implementations, 239
small-step semantics, 221–222
return addresses, 239
revelations (of type identities), 530
reversal of lists, *see* lists, reversal
R^nRS Scheme standards, 168
root(s), 260–261, 290
of absolute access paths (Molecule), 559
in μScheme+, 265
of module types in environments, 563
rooted module types, 563
rules, S97
inference, *see* inference rules
lowering (μScheme+), 214
in Prolog, S45
typing, *see* typing rules
run-time errors, 47
checked, 40
unchecked, 40

S-expressions, 91–92, 174
equality, 104
fully general, 94
functions involving, 103–104
inference rules, defined by, 117
limitations, 457
ordinary, 92
records, encoded with, 107–109
trees, encoded with, 109–110
safety properties, 240, *see also* memory safety, type safety
satisfaction (of type-equality constraints), 428
satisfiability, *see* Boolean satisfiability
satisfying assignment, 138
to logical variables in Prolog, S49
Scheme, *see also* μScheme
continuations, 172
data abstraction in, 588
equality, 633
μScheme vs, 168–172
Smalltalk blocks and, 638
standards, 168
scoping (dynamic, lexical, and static), 148
scrutinees, 458, 459, 468, 502
evaluated in case expressions, 490
sealing, 532, 539
examples, 558–560
in real languages, 580
in the subtype relation, 566
type checking, 564

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

sealing (continued)
type constructors and, 535
search
with default result, 137
higher-order functions for, 127–128
using continuations, 136–143
selection from lists
higher-order functions, 127
μSmalltalk methods, 645
visualizations, 129
selectors, *see* observers
`self`, 615, 711
Self (programming language), 609, 707
semantics, big-step, *see* operational semantics
semantics, denotational, *see* denotational semantics
semantics, operational, *see* operational semantics
semantics, small-step, *see* small-step semantics
semispaces, 271
sentinels
in μSmalltalk lists, 673
separately compiled interfaces
in real languages, 580
sequenceable collections
implementations, 661–662
protocols, 648–649, 659
`SequenceableCollection` protocol, 648
sequencing, *see* `begin`
set(s)
of association lists, 132–135
represented as lists, 104–105, 131
`set`, *see* assignment, *see also* mutable variables
set functions
algebraic laws, 105
polymorphism, 131–133
shape(s), 611–612, 619–622
protocol, 620
Shape, Smalltalk-80 class, 702
shared mutable state, *see* mutable state, shared
`shift` (control operator), 247
shift-reduce parsing, S194
short-circuit Boolean operators, 164–165, 174, 195, 320
side effects, 16, 20, 343
signatures, 528, 581
simple equality constraints, 420
solving, 429–430
Simplified Impcore, 62
Simula 67, 586, 588, 609
inspiration for Smalltalk, 707, 712
simultaneous substitution, 166, 365
single inheritance, 711
Skolem types, S31
slots (in object-oriented languages), 707
small integers, 666
small-step semantics, 71, 210–213, 215–223, 245
function application, 218–220
further reading, 246
`if`, 217–218
`lambda`, 217
`let`, 220
`letrec`, 221
literal expressions, 216
`long-goto`, 221–222
`long-label`, 221–222
metatheoretic proofs and, 244
proof techniques, 240
`return`, 221–222
`set`, 217
tracing, 224–225
uses, 240–242
variables, 217
`SmallInteger`
private methods, 668
protocol, 651
Smalltalk-80, 700–707
binary messages, 703
class hierarchies, 642, 704–707
class instance variables, 704
class variables, 704
in simulations, S140
Collection hierarchy, 705–706
concrete syntax, 702–704
development, 711
images, 701
literals, 701
message cascades, 706
numbers, 705
as operating system, 700–701
precedence, 703
as programming language, 701–704
reflection, 706–707
syntax, 624
`UndefinedObject` class, 654
smart constructors
in binary tries, 513
solvers
Boolean formulas, 139–143
conjunctive normal form
code, 143
laws, 141, 142
type-equality constraints, 418, 428–431
sorting, 101
soundness, S97, *see also* type soundness
of algebraic laws, 65
generalization, 427
Molecule, 578

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

soundness (continued)
proofs, 419
rules for explicit substitutions, 419
of type systems, 350
`type-lambda`, 377–378
Spock, *588*
square brackets
in bridge languages, 117
recommended usage, 117
square roots, 652
Squeak (Smalltalk system), 712
stack(s)
evaluation, 202, 730
in real languages, 239
tracing, 224–225
stack allocation, 290
stack-based evaluation
examples, 212–213
recursive evaluation vs, 211–213
stack frames
abstract syntax, 225–226
evaluation contexts and, 242
μScheme+, 214
μSmalltalk, 678
representation, 223–224
standard libraries, 26, *see also* initial basis
Standard ML, 526
equality, 633
standards, Scheme, 168
statements (syntactic category), 16
static scoping, 148
static typing, 327
sample guarantees, 328
`static` variables (in C), 124
stone knives, *588*
stop-the-world garbage collection, 260
stores, 174
linear, 144
in location semantics, 144
single-threaded, 144, 154
streams, S226–233
for reading from files, S237
strengthening, 559, 570, 571
strict evaluation, 241
struct types, 383
structural equivalence, 383
in type systems, 483
structural induction, 115
structured control-flow constructs, 11
structured operational semantics, *see* operational semantics
structured programming, 201–202
structures (Standard ML form), 527
subclasses, 711
examples, 620–622
subtypes vs, 713
`subclassResponsibility`, 622, 624
examples, 621
subgoals, S97
of a Prolog clause, S52
substitution, 371–373, 409–412, 442, S97, *see also* substitutions
capture-avoiding, *see* capture-avoiding substitution
of equals for equals, 113
evaluation via, 166, 731
quantified types and, 356
simultaneous, 365
syntactic sugar and, 165–167
in translations for μScheme, 162
for type variables, 372
uses, 165–166
substitutions, *see also* substitution
for abstract types, 567
algebraic laws, 409–410
composition, 410–411
creation, 411
explicit, 417
typing judgment, 418
generality, 442, S60
implementations, 410
interpretations, 409, 410
from manifest types, 567
Molecule, 579
most general, 442
type inference, 418–420
for variables, 411
subtype polymorphism, 132, 646
subtyping, 587
behavioral, 627, 708
direction of, 561
module types, 561, 564–566
Molecule, 564–566
rules for, 564
subclassing vs, 713
success continuations, 136, 139
sum types, 465
C, poor support in, 41
rules, 349–350
sums of products, 465–466, 502
in C, 465
in object-oriented languages, 466
programming techniques, 465–466
in Smalltalk, 624
`super`, 711
examples, 621
idiomatic usage, 632–633
messages to, 632
operational semantics, 680
superclasses, 610, 614, 624, 711
`Shape` example, 620
supertypes, direction of, 561
symbol(s) (μScheme values), 92
symbol(s) (μSmalltalk objects), 630

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

`symbol?`, 92
symbol tables, *see* environments
`synerror`, 47
syntactic abstractions, *see* macros
syntactic categories, 16, 71
  Molecule, 539
  types, 330
syntactic forms, 15, 71
syntactic proofs, 55–66
syntactic sugar, 26, 66, 71, 162, 174
  `&&` and `||`, S209
  `begin`, 167
  Boolean operators, 164–165
  `cond`, 163–164
  `define`, 152
    described as, 120
  Impcore, extending with, 66–67
  implementation, 195–197
  `let`, 163
  `let*`, 163
  `letrec`, 163
  in μScheme+, 204–205
  Molecule, 535, 537
  pattern matching, 480–483
  records, 167–168
    Molecule, 542
  short-circuit operators, 164–165
  `while*`, `do-while`, and `for`, 66
syntax, *see* abstract syntax *or* concrete syntax
syntax errors, 47
System F, 385

θ (substitution), 409
tables, *see* finite maps
tail call(s), 170, *see also* optimized tail calls
  in continuation-passing style, 170
  further reading, 246
  implementations, 238–239
  optimized, 170, 214
  proper, 214
tail-call optimization, *see* optimized tail calls
tail position, 253
tail recursion, *see* tail calls
`takewhile`, 206
talent imitates; genius steals, *S245*
term(s), 333, 385, S97
  expressions, as theory-speak for, 333
  in Prolog, S44
term variables, 408
theory (vs metatheory), 59
`throw`, 202, 207–210
  examples, 208–210
  in real languages, 240
throwing an exception, 207
Tiger! Tiger! Tiger!, *56*, *113*, *S13*, *S397*
`TikzCanvas` (μSmalltalk class), 618–619
to-space, 271 (defined)
tokens, lexical, 14, S191
tracing
  message sends, 640, 641
  μScheme calls, 198
  small-step evaluation, 224–225
  Smalltalk, S583
transformation of lists
  higher-order functions, 127
  methods, 645
tree(s)
  encoded as S-expressions, 109–110
  μScheme, 109–110
  traversal(s), 110, 118, 266
  traversal exercises, 181, 515
tree-node functions
  algebraic laws, 109
tricolor marking, 261–262, 290
  in concurrent and generational collectors, 286
  in copying collectors, 273
  examples, 274–276
  in mark-sweep collectors, 269
tries
  exercises, 512–514
trivial constraint, 420
Trotsky, Leon, *136*
`True` (μSmalltalk class), 655
true definitions, 18
`try-catch`, 204, 207–210
  examples, 208–210
  in real languages, 240
tuple types
  algebraic laws, 349
  typing rules, 348
type(s), 385, 442, *see also* type systems
  algebraic notation for, 357
  arrays, 343
  dependent, 727
  elaboration (Molecule), 571–574
  inhabitants, 334
  modules, *see* module types
  Molecule, 535
  nano-ML, 407–409
  notation for, 357
  principal, 416–417
  quantified, 351, 356–360
  representation
    Typed μScheme, 357
  Typed Impcore, 331
  uninhabited, 334
  user-defined, 459–476, 486–490, 502
  uses, 327–328
type abbreviations, 320, 484, 502, 529
  in ML, 303

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

type abstraction, 385, *see also* `type-lambda`
  implicit, 413
  in ML, 412
  typing rules, 365
type annotations, 401
type application, 358, 385, *see also* instantiation
  implicit, 413
  in ML, 412
  typing rules, 365
type ascriptions
  in Standard ML, 441
type checkers, 329, 385, *see also* type checking
  arrays, 348
  μML, 489, 497, 498
  Molecule, S496–518
  nano-ML, 437–439
  Typed Impcore, 338–342
  Typed μScheme, 366–367
type checking, 327, *see also* type checkers
  modular, 558
    in real languages, 580
  Molecule
    definitions, 570–571
type classes, 589
  overloading and, 585
type components, 538
type constructors, 385
  M (set of), 487
  nano-ML, 407, 409
  nullary, 343
  representations (μML), 485
  sealing and, 535
  Typed Impcore, 333
type declarations, manifest, 530
`typedef`, 484
type environments
  nano-ML, 407–409
  Typed Impcore, 333
type equality, *see also* type equivalence
  dire warning about, 332
type-equality constraints, 418, 420–431, 442
  grammar, 420
  solving, 428–431
  typing judgment, 420
type equivalence, 367–370
  Hindley-Milner, 412
  μML, 483–485
  Molecule, 559
  nano-ML, 412
  in real languages, 383
  rules, 369–370
  structural, 483
  Typed Impcore, 332
type equivalence (continued)
  Typed μScheme, 370
type erasure, 380
type-formation rules, 345, 347
  kinds vs, 352–355
  modules, 538–539
  template, 346
  Typed Impcore, 334
type generativity, 483–485, *see also* generativity
type identity
  examples, 559
type inference, 327, 401, 442
  constraints for, 420–428
  decidability, 401
  implementations, 437–439
  performance, 430
  polymorphic types, 423–424
  substitutions for, 418–420
`type-lambda`, 412, *see also* type abstraction
  implicit, 413
  side conditions, 377–378
  soundness, 377–378
  in Typed μScheme, 352, 359
  typing rules, 365
  variable capture and, 377–378
type-level expressions, 352, 357
type parameters, 132, 304, 327, 356
type predicates
  μScheme, 92
  for records, 108
type safety, 328–329
type schemes, 407–409, 442
  canonical, 433
  degenerate, 414
  principal, 417
type soundness, 350–351
type synonyms, *see* type abbreviations
type systems, 327, 385
  design (Molecule), 577–579
  design goals (Molecule), 558
  further reading, 385
  loopholes, 376–377
  μML, 483–486, 495–499
  modules
    further reading, 588
  Molecule, 558–579
    environment lookup, 574–576
    formalism, 561–564
    overloading, 576–577
    overview, 562
    soundness, 578
  nano-ML, 407–417
  in real languages, 383
  restrictive, 328
  soundness, 350–351
  soundness (Molecule), 578

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

type systems (continued)
subversion, 376–377
Typed Impcore, 333–337
Typed μScheme, 352–378
type variables, 320, 356
captured by a naïve instantiation, 373
fresh, 433–434
nano-ML, 407
term variables vs, 408
understanding, 356
typing
of definitions, 341, 366
nondeterministic, 416–417
typing contexts
Molecule, 562
typing judgments
Molecule, 565
nano-ML, 418 (explicit substitutions), 420 (constraints)
Typed Impcore, 335, 336
Typed μScheme, 363
typing rules
arrays, 346–348
arrow types, 348
case expressions, 349, 495–496 (nondeterministic), 497–499 (constraint-based)
functions, 348
μML 495–496 (nondeterministic), 497–499 (constraint-based)
Molecule definitions, 569
nano-ML, 412 (nondeterministic), 416 (nondeterministic), 421 (constraint-based), 428 (constraint-based)
pairs, 348–349
pattern matching, 495–496 (nondeterministic), 497–499 (constraint-based)
products, 348
sums, 349–350, *see also* sum types
tuple types, 348
tuples, 348
Typed Impcore, 334–337
Typed μScheme, 361–366
typical, 348–350
unions, 349–350

μML, alphabetized as micro-ML
unchecked run-time errors, 40
in `getline_`, S165
in printing functions, 46
from unspecified values, 151
uncurried functions, 125
uncurrying, 126
`UndefinedObject` (μSmalltalk class), 630, 696
undelimited continuations, 243, 245
understood
message in Smalltalk, 631
unfold, S230
Unicode characters (μSmalltalk), 641
Unicode code points, 23
unification, 417, 442, S97
in type inference, 417
unifiers, 418
uninhabited module types
examples, 566
uninhabited types, 334
union types
rules, 349–350
unit tests, 24–26
unit type, 334
unreachability, *see* reachability
unsafe functions
for printing, 46
unsafe language features
uses, 328–329
unspecified values, 146
in μScheme, 151
misuse by careless persons, S328
in Typed μScheme, 380
unwinding (call stacks), 583
μScheme, alphabetized as micro-Scheme
`use` (extended-definition form), 23, 25
user-defined functions
in Impcore environment ϕ, 30
user-defined types, 459–476, 486–490, 502
μSmalltalk, alphabetized as micro-Smalltalk

`val`, 22
operational semantics
Impcore, 38
μScheme, 151
μSmalltalk, 685
semantic comparison, 402
typing rules
nano-ML, 416 (nondeterministic), 423 (constraint-based)
Typed Impcore, 337
Typed μScheme, 366
`val-rec`, 402
rationale, 362
typing rules
nano-ML, 416 (nondeterministic)
Typed μScheme, 366
valid derivations, 57–58
value(s)
μML, 468, 486
μScheme, 91–92, 120, 145
μSmalltalk, 629–630
in μSmalltalk semantics, 677

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-107-18018-5 — Programming Languages: Build, Prove, and Compare
Norman Ramsey
Index
More Information

value(s) (continued)
Molecule, 535
nano-ML, 405
representations
μScheme (C), 152–153
μScheme (ML), 306
μSmalltalk, 685–686
Typed Impcore, 332
unspecified, 146
value components, 538
value constructors, 320, 457, 466, 502
in Erlang, 499
misspelled, 470
in Molecule, 533
names usable for abstract syntax, 42
in patterns, 458
polymorphic, 462–463
Prolog equivalent of, S52
representations (μML), 490
value variables vs, 468–471
value restriction
exercises, 452
in ML, S81
value semantics, 174, 405–407
value variables, 466
value constructors vs, 468–471
"varargs", *see* variadic functions
variable(s), *see also* metavariables, program variables, term variables, type variables, value variables
global, 151–152
logical, *see* logical variables
metavariables vs, 19
operational semantics
Impcore, 32
μScheme, 145
μSmalltalk, 680
small-step semantics, 217
Smalltalk-80 meaning, 614
typing rules
nano-ML, 413 (nondeterministic), 422 (constraint-based)
Typed Impcore, 335
variable(s) (continued)
typing rules (continued)
Typed μScheme, 363
value constructors vs, 468–471
variable capture, 373
in desugaring for `||`, 165
of type variables, 373
`type-lambda` and, 365, 377–378
variadic functions, 169, 322 (exercise), S177
for extensible printers, S177
verification
of invariants, 553
visibility
abstract types vs objects, 626
instance variables, 624
μSmalltalk, 631
to μSmalltalk methods, 615
Molecule, 631
objects vs abstract types, 626
visualizations
backtracking, 139
list operations, 129

weakest preconditions, 246
wedge ($\wedge$, type intersection), 564
`while`
operational semantics
Impcore, 35
μScheme, 149
typing rules
Typed Impcore, 336
Typed μScheme, 363
`WHILEX` vs `WHILE`, 42
white objects, 261
in copying collection, 273
in mark-sweep collection, 269
wholemeal programming, 172
wildcard patterns
in μML, 461
in ML, 307
write barrier, 286
wrong, *see* going wrong

Xerox PARC, 711

zippers, 510–512 (exercise)

© in this web service Cambridge University Press & Assessment www.cambridge.org