Appendix A: The bare Interpreter (Version 1)

```
(* interp1.sml:   Mini ML interpreter, VERSION 1 *)

signature INTERPRETER=
   sig
      val interpret: string -> string
      val eval: bool ref
      and tc  : bool ref
   end;

                (* syntax *)

signature EXPRESSION =
   sig
      datatype Expression =
         SUMexpr of Expression * Expression   |
         DIFFexpr of Expression * Expression   |
         PRODexpr of Expression * Expression   |
         BOOLexpr of bool   |
         EQexpr of Expression * Expression   |
         CONDexpr of Expression * Expression * Expression   |
         CONSexpr of Expression * Expression   |
         LISTexpr of Expression list   |
         DECLexpr of string * Expression * Expression   |
         RECDECLexpr of string * Expression * Expression   |
         IDENTexpr of string   |
         LAMBDAexpr of string * Expression   |
         APPLexpr of Expression * Expression   |
         NUMBERexpr of int
   end


            (* parsing *)

signature PARSER =
   sig
      structure E: EXPRESSION

      exception Lexical of string
      exception Syntax of string

      val parse: string -> E.Expression
   end
```

```
                        (* environments *)

signature ENVIRONMENT =
   sig
      type 'object Environment

      exception Retrieve of string

      val emptyEnv: 'object Environment
      val declare: string * 'object * 'object Environment ->
                                        'object Environment
      val retrieve: string * 'object Environment -> 'object
   end

                        (* evaluation *)
signature VALUE =
   sig
      type Value
      exception Value

      val mkValueNumber: int -> Value
          and unValueNumber: Value -> int

      val mkValueBool: bool -> Value
          and unValueBool: Value -> bool

      val ValueNil: Value
      val mkValueCons: Value * Value -> Value
          and unValueHead: Value -> Value
          and unValueTail: Value -> Value

      val eqValue: Value * Value -> bool
      val printValue: Value -> string
   end


signature EVALUATOR =
   sig
      structure Exp: EXPRESSION
      structure Val: VALUE
      exception Unimplemented
      val evaluate: Exp.Expression -> Val.Value
   end

                  (* type checking *)
```

```
signature TYPE =
   sig
      type Type

(*constructors and decstructors*)
      exception Type
      val mkTypeInt: unit -> Type
          and unTypeInt: Type -> unit

      val mkTypeBool: unit -> Type
          and unTypeBool: Type -> unit

      val prType: Type->string
   end



signature TYPECHECKER =
   sig
      structure Exp: EXPRESSION
      structure Type: TYPE
      exception NotImplemented of string
      exception TypeError of Exp.Expression * string
      val typecheck: Exp.Expression -> Type.Type
   end;

                  (* the interpreter*)

functor Interpreter
   (structure Ty: TYPE
    structure Value : VALUE
    structure Parser: PARSER
    structure TyCh: TYPECHECKER
    structure Evaluator:EVALUATOR
      sharing Parser.E = TyCh.Exp = Evaluator.Exp
          and TyCh.Type = Ty
          and Evaluator.Val = Value
   ): INTERPRETER=

struct
  val eval= ref false   (* toggle for evaluation *)
  and tc  = ref true    (* toggle for type checking *)
  fun interpret(str)=
    let val abstsyn= Parser.parse str
        val typestr= if !tc then Ty.prType(TyCh.typecheck abstsyn)
```

```
                    else "(disabled)"
        val valuestr= if !eval then
                        Value.printValue(Evaluator.evaluate abstsyn)
                      else "(disabled)"

    in  valuestr ^ " : " ^ typestr
    end
    handle Evaluator.Unimplemented =>
              "Evaluator not fully implemented"
         | TyCh.NotImplemented msg =>
              "Type Checker not fully implemented " ^ msg
         | Value.Value    => "Run-time error"
         | Parser.Syntax msg => "Syntax Error: " ^ msg
         | Parser.Lexical msg=> "Lexical Error: " ^ msg
         | TyCh.TypeError(_,msg)=> "Type Error: " ^ msg
end;

                    (* the evaluator *)

functor Evaluator
  (structure Expression: EXPRESSION
   structure Value: VALUE):EVALUATOR=

   struct
      structure Exp= Expression
      structure Val= Value
      exception Unimplemented

      local
         open Expression Value
         fun evaluate exp =
            case exp
              of BOOLexpr b => mkValueBool b
               | NUMBERexpr i => mkValueNumber i
               | SUMexpr(e1, e2) =>
                    let val e1' = evaluate e1
                        val e2' = evaluate e2
                    in
                       mkValueNumber(unValueNumber e1'
                                  + unValueNumber e2')
                    end

               | DIFFexpr(e1, e2) =>
                    let val e1' = evaluate e1
                        val e2' = evaluate e2
```

```
                    in
                       mkValueNumber(unValueNumber e1'
                                   - unValueNumber e2')
                    end

                | PRODexpr(e1, e2) =>
                    let val e1' = evaluate e1
                        val e2' = evaluate e2
                    in
                       mkValueNumber(unValueNumber e1'
                                   * unValueNumber e2')
                    end

                | EQexpr _ => raise Unimplemented
                | CONDexpr _ => raise Unimplemented
                | CONSexpr _ => raise Unimplemented
                | LISTexpr _ => raise Unimplemented
                | DECLexpr _ => raise Unimplemented
                | RECDECLexpr _ => raise Unimplemented
                | IDENTexpr _ => raise Unimplemented
                | LAMBDAexpr _ => raise Unimplemented
                | APPLexpr _ => raise Unimplemented

      in
         val evaluate = evaluate
      end
   end;

                        (* the type checker *)

functor TypeChecker
  (structure Ex: EXPRESSION
   structure Ty: TYPE)=
struct
  structure Exp = Ex
  structure Type = Ty
  exception NotImplemented of string
  exception TypeError of Ex.Expression * string

  fun tc (exp: Ex.Expression): Ty.Type =
    case exp of
      Ex.BOOLexpr b => raise NotImplemented "(bool const)"
    | Ex.NUMBERexpr _ => Ty.mkTypeInt()
    | Ex.SUMexpr(e1,e2)  => checkIntBin(e1,e2)
    | Ex.DIFFexpr(e1,e2) => raise NotImplemented "(minus)"
```

```
    | Ex.PRODexpr(e1,e2) => raise NotImplemented "(multiplication)"
    | Ex.LISTexpr _ => raise NotImplemented "(lists)"
    | Ex.CONSexpr _ => raise NotImplemented "(lists)"
    | Ex.EQexpr _ => raise NotImplemented "(equality)"
    | Ex.CONDexpr _ => raise NotImplemented "(conditional)"
    | Ex.DECLexpr _ => raise NotImplemented "(declaration)"
    | Ex.RECDECLexpr _ => raise NotImplemented "(rec decl)"
    | Ex.IDENTexpr _   => raise NotImplemented "(identifier)"
    | Ex.LAMBDAexpr _  => raise NotImplemented "(function)"
    | Ex.APPLexpr _ => raise NotImplemented    "(application)"


  and checkIntBin(e1,e2) =
    let val t1 = tc e1
        val _  = Ty.unTypeInt t1
                 handle Ty.Type=> raise TypeError(e1,
                                          "expected int")
        val t2 = tc e2
        val _  = Ty.unTypeInt t2
                 handle Ty.Type=> raise TypeError(e2,
                                          "expected int")
     in Ty.mkTypeInt()
    end;

  val typecheck = tc

end; (*TypeChecker*)



                    (* the basics -- nullary functors *)

functor Type():TYPE =
struct
  datatype Type = INT
                | BOOL

  exception Type

  fun mkTypeInt() = INT
  and unTypeInt(INT)=()
    | unTypeInt(_)= raise Type

  fun mkTypeBool() = BOOL
  and unTypeBool(BOOL)=()
```

```
    | unTypeBool(_)= raise Type

  fun prType INT = "int"
  |   prType BOOL= "bool"
end;



functor Expression(): EXPRESSION =
   struct
      type 'a pair = 'a * 'a

      datatype Expression =
         SUMexpr of Expression pair   |
         DIFFexpr of Expression pair   |
         PRODexpr of Expression pair   |
         BOOLexpr of bool   |
         EQexpr of Expression pair   |
         CONDexpr of Expression * Expression * Expression   |
         CONSexpr of Expression pair   |
         LISTexpr of Expression list   |
         DECLexpr of string * Expression * Expression   |
         RECDECLexpr of string * Expression * Expression   |
         IDENTexpr of string   |
         LAMBDAexpr of string * Expression   |
         APPLexpr of Expression * Expression   |
         NUMBERexpr of int
   end;

functor Value(): VALUE =
   struct
      type 'a pair = 'a * 'a

      datatype Value = NUMBERvalue of int   |
                       BOOLvalue of bool   |
                       NILvalue   |
                       CONSvalue of Value pair

      exception Value

      val mkValueNumber = NUMBERvalue
      val mkValueBool = BOOLvalue

      val ValueNil = NILvalue
      val mkValueCons = CONSvalue
```

```
    fun unValueNumber(NUMBERvalue(i)) = i   |
        unValueNumber(_) = raise Value

    fun unValueBool(BOOLvalue(b)) = b   |
        unValueBool(_) = raise Value

    fun unValueHead(CONSvalue(c, _)) = c   |
        unValueHead(_) = raise Value

    fun unValueTail(CONSvalue(_, c)) = c   |
        unValueTail(_) = raise Value

    fun eqValue(c1, c2) = (c1 = c2)

(* Pretty-printing *)

    fun intToString(i:int)=  (if i<0 then " -" else "")^
                                    natToString (abs i)
    and natToString(n:int)=
        let val d = n div 10 in
          if d = 0 then chr(ord"0" + n)
          else natToString(d)^ chr(ord"0" + (n mod 10))
        end
    fun printValue(NUMBERvalue(i)) = intToString(i)   |
        printValue(BOOLvalue(true)) = "true"   |
        printValue(BOOLvalue(false)) = "false"   |
        printValue(NILvalue) = "[]"   |
        printValue(CONSvalue(cons)) =
                        "[" ^ printValueList(cons) ^ "]"
        and printValueList(hd, NILvalue) = printValue(hd)   |
            printValueList(hd, CONSvalue(tl)) =
               printValue(hd) ^ ", " ^ printValueList(tl)   |
            printValueList(_) = raise Value
end;
```

Appendix B: Files

The following files are available:

Part 1

- `examples.sml` The examples from Part 1

Part 2

- `interp1.sml` Version 1 (as included in Appendix A).
- `interp2.sml` ··· `interp4.sml` The other versions.
- `build1.sml` the structure declarations needed to build Version 1.
- `build2.sml` ··· `build4.sml` Similarly for the other versions.
- `parser.sml` The parser functor.

To build Version 3, say, you type the following (assuming you have copied the files to your directory):

```
use "interp3.sml";
use "parser.sml";
use "build3.sml";
```

Since the parser functor is completely closed, you don't have to include it more than once in every session, although you will probably want to build your system several times while you experiment with the extensions.

Lecture Notes in Computer Science

For information about Vols. 1–1054

please contact your bookseller or Springer-Verlag

Vol. 1055: T. Margaria, B. Steffen (Eds.), Tools and Algorithms for the Construction and Analysis of Systems. Proceedings, 1996. XI, 435 pages. 1996.

Vol. 1056: A. Haddadi, Communication and Cooperation in Agent Systems. XIII, 148 pages. 1996. (Subseries LNAI).

Vol. 1057: P. Apers, M. Bouzeghoub, G. Gardarin (Eds.), Advances in Database Technology — EDBT '96. Proceedings, 1996. XII, 636 pages. 1996.

Vol. 1058: H. R. Nielson (Ed.), Programming Languages and Systems – ESOP '96. Proceedings, 1996. X, 405 pages. 1996.

Vol. 1059: H. Kirchner (Ed.), Trees in Algebra and Programming – CAAP '96. Proceedings, 1996. VIII, 331 pages. 1996.

Vol. 1060: T. Gyimóthy (Ed.), Compiler Construction. Proceedings, 1996. X, 355 pages. 1996.

Vol. 1061: P. Ciancarini, C. Hankin (Eds.), Coordination Languages and Models. Proceedings, 1996. XI, 443 pages. 1996.

Vol. 1062: E. Sanchez, M. Tomassini (Eds.), Towards Evolvable Hardware. IX, 265 pages. 1996.

Vol. 1063: J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, D. Snyers (Eds.), Artificial Evolution. Proceedings, 1995. XIII, 396 pages. 1996.

Vol. 1064: B. Buxton, R. Cipolla (Eds.), Computer Vision – ECCV '96. Volume I. Proceedings, 1996. XXI, 725 pages. 1996.

Vol. 1065: B. Buxton, R. Cipolla (Eds.), Computer Vision – ECCV '96. Volume II. Proceedings, 1996. XXI, 723 pages. 1996.

Vol. 1066: R. Alur, T.A. Henzinger, E.D. Sontag (Eds.), Hybrid Systems III. IX, 618 pages. 1996.

Vol. 1067: H. Liddell, A. Colbrook, B. Hertzberger, P. Sloot (Eds.), High-Performance Computing and Networking. Proceedings, 1996. XXV, 1040 pages. 1996.

Vol. 1068: T. Ito, R.H. Halstead, Jr., C. Queinnec (Eds.), Parallel Symbolic Languages and Systems. Proceedings, 1995. X, 363 pages. 1996.

Vol. 1069: J.W. Perram, J.-P. Müller (Eds.), Distributed Software Agents and Applications. Proceedings, 1994. VIII, 219 pages. 1996. (Subseries LNAI).

Vol. 1070: U. Maurer (Ed.), Advances in Cryptology – EUROCRYPT '96. Proceedings, 1996. XII, 417 pages. 1996.

Vol. 1071: P. Miglioli, U. Moscato, D. Mundici, M. Ornaghi (Eds.), Theorem Proving with Analytic Tableaux and Related Methods. Proceedings, 1996. X, 330 pages. 1996. (Subseries LNAI).

Vol. 1072: R. Kasturi, K. Tombre (Eds.), Graphics Recognition. Proceedings, 1995. X, 308 pages. 1996.

Vol. 1073: J. Cuny, H. Ehrig, G. Engels, G. Rozenberg (Eds.), Graph Grammars and Their Application to Computer Science. Proceedings, 1994. X, 565 pages. 1996.

Vol. 1074: G. Dowek, J. Heering, K. Meinke, B. Möller (Eds.), Higher-Order Algebra, Logic, and Term Rewriting. Proceedings, 1995. VII, 287 pages. 1996.

Vol. 1075: D. Hirschberg, G. Myers (Eds.), Combinatorial Pattern Matching. Proceedings, 1996. VIII, 392 pages. 1996.

Vol. 1076: N. Shadbolt, K. O'Hara, G. Schreiber (Eds.), Advances in Knowledge Acquisition. Proceedings, 1996. XII, 371 pages. 1996. (Subseries LNAI).

Vol. 1077: P. Brusilovsky, P. Kommers, N. Streitz (Eds.), Mulimedia, Hypermedia, and Virtual Reality. Proceedings, 1994. IX, 311 pages. 1996.

Vol. 1078: D.A. Lamb (Ed.), Studies of Software Design. Proceedings, 1993. VI, 188 pages. 1996.

Vol. 1079: Z.W. Raś, M. Michalewicz (Eds.), Foundations of Intelligent Systems. Proceedings, 1996. XI, 664 pages. 1996. (Subseries LNAI).

Vol. 1080: P. Constantopoulos, J. Mylopoulos, Y. Vassiliou (Eds.), Advanced Information Systems Engineering. Proceedings, 1996. XI, 582 pages. 1996.

Vol. 1081: G. McCalla (Ed.), Advances in Artificial Intelligence. Proceedings, 1996. XII, 459 pages. 1996. (Subseries LNAI).

Vol. 1082: N.R. Adam, B.K. Bhargava, M. Halem, Y. Yesha (Eds.), Digital Libraries. Proceedings, 1995. Approx. 310 pages. 1996.

Vol. 1083: K. Sparck Jones, J.R. Galliers, Evaluating Natural Language Processing Systems. XV, 228 pages. 1996. (Subseries LNAI).

Vol. 1084: W.H. Cunningham, S.T. McCormick, M. Queyranne (Eds.), Integer Programming and Combinatorial Optimization. Proceedings, 1996. X, 505 pages. 1996.

Vol. 1085: D.M. Gabbay, H.J. Ohlbach (Eds.), Practical Reasoning. Proceedings, 1996. XV, 721 pages. 1996. (Subseries LNAI).

Vol. 1086: C. Frasson, G. Gauthier, A. Lesgold (Eds.), Intelligent Tutoring Systems. Proceedings, 1996. XVII, 688 pages. 1996.

Vol. 1087: C. Zhang, D. Lukose (Eds.), Distributed Artificial Intelligence. Proceedings, 1995. VIII, 232 pages. 1996. (Subseries LNAI).

Vol. 1088: A. Strohmeier (Ed.), Reliable Software Technologies – Ada-Europe '96. Proceedings, 1996. XI, 513 pages. 1996.

Vol. 1089: G. Ramalingam, Bounded Incremental Computation. XI, 190 pages. 1996.

Vol. 1090: J.-Y. Cai, C.K. Wong (Eds.), Computing and Combinatorics. Proceedings, 1996. X, 421 pages. 1996.

Vol. 1091: J. Billington, W. Reisig (Eds.), Application and Theory of Petri Nets 1996. Proceedings, 1996. VIII, 549 pages. 1996.

Vol. 1092: H. Kleine Büning (Ed.), Computer Science Logic. Proceedings, 1995. VIII, 487 pages. 1996.

Vol. 1093: L. Dorst, M. van Lambalgen, F. Voorbraak (Eds.), Reasoning with Uncertainty in Robotics. Proceedings, 1995. VIII, 387 pages. 1996. (Subseries LNAI).

Vol. 1094: R. Morrison, J. Kennedy (Eds.), Advances in Databases. Proceedings, 1996. XI, 234 pages. 1996.

Vol. 1095: W. McCune, R. Padmanabhan, Automated Deduction in Equational Logic and Cubic Curves. X, 231 pages. 1996. (Subseries LNAI).

Vol. 1096: T. Schäl, Workflow Management Systems for Process Organisations. XII, 200 pages. 1996.

Vol. 1097: R. Karlsson, A. Lingas (Eds.), Algorithm Theory – SWAT '96. Proceedings, 1996. IX, 453 pages. 1996.

Vol. 1098: P. Cointe (Ed.), ECOOP '96 – Object-Oriented Programming. Proceedings, 1996. XI, 502 pages. 1996.

Vol. 1099: F. Meyer auf der Heide, B. Monien (Eds.), Automata, Languages and Programming. Proceedings, 1996. XII, 681 pages. 1996.

Vol. 1100: B. Pfitzmann, Digital Signature Schemes. XVI, 396 pages. 1996.

Vol. 1101: M. Wirsing, M. Nivat (Eds.), Algebraic Methodology and Software Technology. Proceedings, 1996. XII, 641 pages. 1996.

Vol. 1102: R. Alur, T.A. Henzinger (Eds.), Computer Aided Verification. Proceedings, 1996. XII, 472 pages. 1996.

Vol. 1103: H. Ganzinger (Ed.), Rewriting Techniques and Applications. Proceedings, 1996. XI, 437 pages. 1996.

Vol. 1104: M.A. McRobbie, J.K. Slaney (Eds.), Automated Deduction – CADE-13. Proceedings, 1996. XV, 764 pages. 1996. (Subseries LNAI).

Vol. 1105: T.I. Ören, G.J. Klir (Eds.), Computer Aided Systems Theory – CAST '94. Proceedings, 1994. IX, 439 pages. 1996.

Vol. 1106: M. Jampel, E. Freuder, M. Maher (Eds.), Over-Constrained Systems. X, 309 pages. 1996.

Vol. 1107: J.-P. Briot, J.-M. Geib, A. Yonezawa (Eds.), Object-Based Parallel and Distributed Computation. Proceedings, 1995. X, 349 pages. 1996.

Vol. 1108: A. Díaz de Ilarraza Sánchez, I. Fernández de Castro (Eds.), Computer Aided Learning and Instruction in Science and Engineering. Proceedings, 1996. XIV, 480 pages. 1996.

Vol. 1109: N. Koblitz (Ed.), Advances in Cryptology – Crypto '96. Proceedings, 1996. XII, 417 pages. 1996.

Vol. 1110: O. Danvy, R. Glück, P. Thiemann (Eds.), Partial Evaluation. Proceedings, 1996. XII, 514 pages. 1996.

Vol. 1111: J.J. Alferes, L. Moniz Pereira, Reasoning with Logic Programming. XXI, 326 pages. 1996. (Subseries LNAI).

Vol. 1112: C. von der Malsburg, W. von Seelen, J.C. Vorbrüggen, B. Sendhoff (Eds.), Artificial Neural Networks – ICANN 96. Proceedings, 1996. XXV, 922 pages. 1996.

Vol. 1113: W. Penczek, A. Szałas (Eds.), Mathematical Foundations of Computer Science 1996. Proceedings, 1996. X, 592 pages. 1996.

Vol. 1114: N. Foo, R. Goebel (Eds.), PRICAI'96: Topics in Artificial Intelligence. Proceedings, 1996. XXI, 658 pages. 1996. (Subseries LNAI).

Vol. 1115: P.W. Eklund, G. Ellis, G. Mann (Eds.), Conceptual Structures: Knowledge Representation as Interlingua. Proceedings, 1996. XIII, 321 pages. 1996. (Subseries LNAI).

Vol. 1116: J. Hall (Ed.), Management of Telecommunication Systems and Services. XXI, 229 pages. 1996.

Vol. 1117: A. Ferreira, J. Rolim, Y. Saad, T. Yang (Eds.), Parallel Algorithms for Irregularly Structured Problems. Proceedings, 1996. IX, 358 pages. 1996.

Vol. 1118: E.C. Freuder (Ed.), Principles and Practice of Constraint Programming — CP 96. Proceedings, 1996. XIX, 574 pages. 1996.

Vol. 1119: U. Montanari, V. Sassone (Eds.), CONCUR '96: Concurrency Theory. Proceedings, 1996. XII, 751 pages. 1996.

Vol. 1120: M. Deza, R. Euler, I. Manoussakis (Eds.), Combinatorics and Computer Science. Proceedings, 1995. IX, 415 pages. 1996.

Vol. 1121: P. Perner, P. Wang, A. Rosenfeld (Eds.), Advances in Structural and Syntactical Pattern Recognition. Proceedings, 1996. X, 393 pages. 1996.

Vol. 1122: H. Cohen (Ed.), Algorithmic Number Theory. Proceedings, 1996. IX, 405 pages. 1996.

Vol. 1123: L. Bougé, P. Fraigniaud, A. Mignotte, Y. Robert (Eds.), Euro-Par'96. Parallel Processing. Proceedings, 1996, Vol. I. XXXIII, 842 pages. 1996.

Vol. 1124: L. Bougé, P. Fraigniaud, A. Mignotte, Y. Robert (Eds.), Euro-Par'96. Parallel Processing. Proceedings, 1996, Vol. II. XXXIII, 926 pages. 1996.

Vol. 1125: J. von Wright, J. Grundy, J. Harrison (Eds.), Theorem Proving in Higher Order Logics. Proceedings, 1996. VIII, 447 pages. 1996.

Vol. 1126: J.J. Alferes, L. Moniz Pereira, E. Orlowska (Eds.), Logics in Artificial Intelligence. Proceedings, 1996. IX, 417 pages. 1996. (Subseries LNAI).

Vol. 1127: L. Böszörményi (Ed.), Parallel Computation. Proceedings, 1996. XI, 235 pages. 1996.

Vol. 1128: J. Calmet, C. Limongelli (Eds.), Design and Implementation of Symbolic Computation Systems. Proceedings, 1996. IX, 356 pages. 1996.

Vol. 1129: J. Launchbury, E. Meijer, T. Sheard (Eds.), Advanced Functional Programming. Proceedings, 1996. VII, 238 pages. 1996.