[A Preliminary Work]

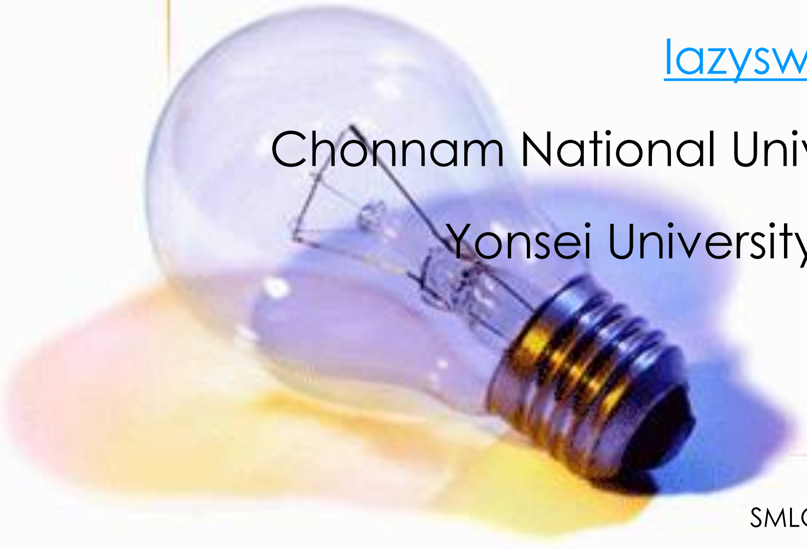
SMLOG: an Embedding of Logic Language in Standard ML

Kwanghoon Choi

lazyswamp@gmail.com

Chonnam National University, Gwangju, Korea (Sept. 1st~)

Yonsei University, Wonju, Korea (~Aug. 31st)



Example 1

- Logic programming
  - E.g., path (“a”, “b”, P).

```
pre edge : (string, string)
pre edge ("a", "b").
    edge ("a", "d").
    edge ("b", "c").
    edge ("b", "d").
    edge ("c", "d").
    edge ("c", "e").
    edge ("d", "e").

pre path : (string, string, string list)
pre path (X, X, [X]).
    path (X, Z, X::Nodes) = edge (X,Y), path (Y,Z,Nodes).
```

Example 2

- Calling SML functions from SMLOG
- User-defined data types
  - E.g., to print stars as the user-defined number

```
fun print : (string : in)

datatype num = Zero | Succ of num

pre prnum : (num)
pre prnum (Zero) = print ("*\\n").
    prnum (Succ D) = print ("*"), prnum (D).
```

Example 3: More Complex Example

```
(* Temporal lambda calculus *)

•    datatype temporalty =
          TInt | TFun of temporalty * temporalty | TO   of temporalty

     datatype temporalterm =
          TVar of string
         | TLam of string * temporalty * temporalterm
         | TApp of temporalterm * temporalterm
         | TNext of temporalterm
         | TPrev of temporalterm

     datatype time = Z | S of time

     type temporaltypingenv = (string * temporalty * time) list

     pre  member : (temporaltypingenv, string, temporalty, time)
     pre  member ( (X,T,N) :: E, X, T, N).
          member ( _ :: E, X, T, N) = member (E, X, T, N).
```

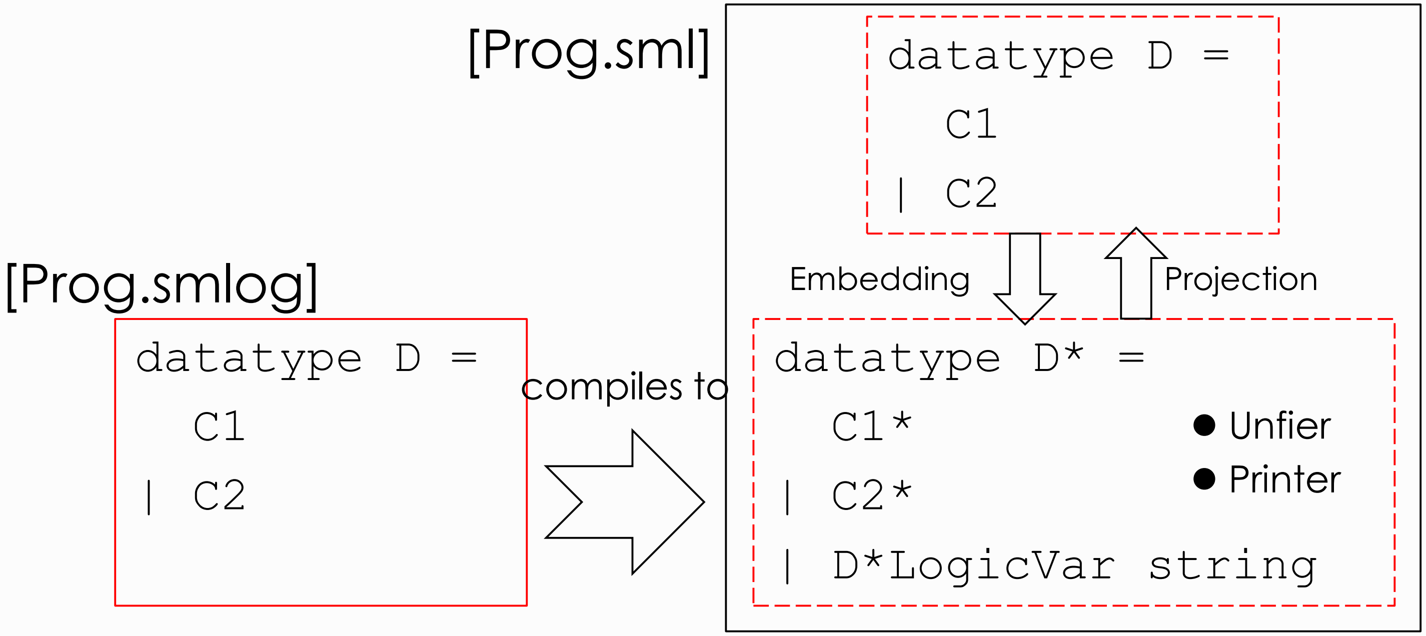
Example 3 (cont.)

(* Temporal lambda calculus (cont.) *)

-
```
pre temporaltyping : (temporaltypingenv, temporalterm, temporalty, time)
pre temporaltyping (E, TVar X, T, N)
      = member (E, X, T, N).

     temporaltyping (E, TLam (X,T,M), TFun (T,T'), N)
      = temporaltyping ((X,T,N) :: E, M, T', N).

     temporaltyping (E, TApp (M1,M2), T, N)
      = temporaltyping (E, M1, TFun (T2,T), N),
        temporaltyping (E, M2, T2, N).

     temporaltyping (E, TNext M, TO T, N)
      = temporaltyping (E, M, T, S N).

     temporaltyping (E, TPrev M, T, S N)
      = temporaltyping (E, M, TO T, N).
```

SMLOG: Terms

- Key Idea 1: Automatically generated unifiers over user-defined datatypes



```
datatype D =
   C1
 | C2
```

```
datatype D* =
   C1*
 | C2*
 | D*LogicVar string
```

●P. Jansson & J. Jeuring, Polytypic unification, JFL1998.

SMLOG: Predicates

- Key Idea 2: Functional interpretation of Logic Programs
  - Predicates are compiled into functions of type “Answer -> Stream Answer”
  - The functions work with a common search model supporting Depth-FS and Breadth-FS

- S. Seres, The Algebra of Logic Programming, Ph.D. Thesis, Oxford Univ., 2001.
- S. Seres & M. Spivey, C.A.R. Hoare, Algebra of Logic Programming, ICLP1999.
- S. Seres & M. Spivey, Embedding Prolog in Haskell, Haskell1998.

Discussion: Calling SMLOG predicates from SML

Compiling

```
- Main.compile “mysort.smlog”;  // Compiling to mysort.sml
val it =() : unit
```

Loading

```
- use “smlogdfs.sml”;     // Loading “struct SMLOG”
- fun le (x,y) = x<=y;   // Defining a comparison function le
val le =fn : int * int -> bool
- use “mysort.sml”;  // Loading the compiled SMLOG program
```

Executing

```
- sort (SOME [9,1,5,4], NONE);   // Calling an SMLOG predicate sort from SML
                                 // through a generated interface fun. named sort
val it = Cons ((SOME [9,1,5,4], SOME [1,4,5,9]), fn)  // The (first) answer
    : (int list option * int list option) SMLOG.stream

- SMLOG.next it;   // Querying the next answer, resulting in no more.
val it = Nil : (int list option * int list option) SMLOG.stream
```

Discussion

DMZ: DeMilitarized Zone

- Any better methods to call SMLOG predicates (& to examine their answers) from SML?
  - *DMZ functions* inside the SMLOG program
- Any methods for programmers to define arbitrary search strategies (beyond DFS & BFS)?

cf. T. Schrijvers & P. Stuckey & P. Wadler, Mondaic Constraint Programming, JFL2009.

- Handling name binders and higher-order functions in SMLOG

Applications

- Programming for fun. & logic
- Test data generation
    - Test data with structural constraints
- Logical framework
    - Environment for prototyping type systems, operational semantics, and so on

Summary: SMLOG

http://github.com/kwanghoon/smlog/

- A deep embedding of a logic programming language such as Prolog into SML
- It aims at Standard ML extended with prolog-like constructs.
- It allows to manipulate SML values using horn-clauses.
- It automatically supports unification over values of user-defined SML datatypes.
- It allows to choose one of DFS and BFS strategies.