جامعة كارنيجي ميلون في قطر
**Carnegie Mellon**
**QATAR CAMPUS**

مؤسسة قطر
*Qatar Foundation*

# On Teaching Programming Languages Using a Wiki

Iliano Cervesato

Carnegie Mellon University – Qatar

**iliano@cmu.edu**

# CMU-CS 15-212
## *"Principles of Programming"*

➢ Sophomore-level CS course
- Advanced programming concepts and skills
- Introduced in the early 1990s

➢ Now, little supporting material
- Notes taken in class
- Few handouts
- Code posted on the web page
- **No book!**
  - Out-of-print or obsolete

➢ A challenge for many students

Put the material on a wiki

# What's a wiki, again?

*…the stuff of Wikipedia*

➢ Collaborative framework to create (and share) information

- Simple, transparent editing
  - Supports text, images, math, sounds, …
- Topic-oriented
  - Short articles (compared to book chapter)
  - Related topics accessible via links
- Collaborative
  - Everybody can be an author
  - Mechanisms to avoid abuse

# The 15-212 Wiki

- Put the whole course on a wiki
  - Categorizes and cross-references material
    - Detailed explanations of material covered in class
    - Lots of examples, exercises
    - Further readings
    - Pointers to advanced material
  - 25% so far (Nov 07 – Jan 08)
  - Built on MediaWiki (same as Wikipedia)
- Semi-open for editing
  - 15-212Q staff and students
    - Create, correct, improve articles
  - Experts, instructors, students elsewhere
    - Upon authorization

*Very preliminary*

1.5 weeks of material

# Sample

Code

Graphics

Prose

Formulas

[edit] Another Example: Reversing a List

The use of an accumulator to make a function tail-recursive is a fairly general technique and applies well beyond integers. We will now see how we can implement list reversal very efficiently in this way.
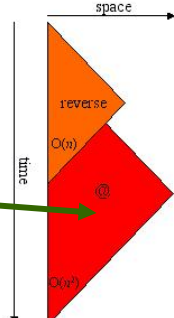
[edit] Naive Implementation

The easy way to reverse a non-empty list is to recursively take its first element and append it to the very end of the result. This is what the function reverse below does.

```
(*
    val reverse : 'a list -> 'a list
        reverse(l) returns a list consisting of the elements of l in reverse order.
    Invariants: none
    Effects: none
*)
fun reverse (nil: 'a list): 'a list = nil
  | reverse (x::l) = (reverse l) @ [x];
```

Because the append function recurses on its first argument, the recursive call is extremely inefficient. Observe the following evaluation trace:

$$reverse\ [2,3,4,5] \Rightarrow (reverse\ [3,4,5]) @ [2]$$
$$\Rightarrow ((reverse\ [4,5]) @ [3]) @ [2]$$
$$\Rightarrow (((reverse\ [5]) @ [4]) @ [3]) @ [2]$$
$$\Rightarrow ((((reverse\ []) @ [5]) @ [4]) @ [3]) @ [2]$$
$$\Rightarrow (((([] @ [5]) @ [4]) @ [3]) @ [2]$$
$$\Rightarrow (((([5] @ [4]) @ [3]) @ [2]$$
$$\Rightarrow ((5::([] @ [4])) @ [3]) @ [2]$$
$$\Rightarrow (([5,4] @ [3]) @ [2]$$
$$\Rightarrow (5::([4] @ [3])) @ [2]$$
$$\Rightarrow (5::4::([] @ [3])) @ [2]$$
$$\Rightarrow [5,4,3] @ [2]$$
$$\Rightarrow 5::([4,3] @ [2])$$
$$\Rightarrow 5::4::([3] @ [2])$$
$$\Rightarrow 5::4::3::([] @ [2])$$
$$\Rightarrow [5,4,3,2]$$

space

reverse

$O(n)$

time

@

$O(n^2)$

The unfolding of reverse leaves an expression to be evaluate that contains 4 calls to append, each as the left argument of the next. Evaluating this expression processes the leftmost element of the resulting list (here 5) over and over. Indeed, it can be shown that the reversal of a list of $n$ elements involves the creation of $n(n+1)/2$ list elements (as witnessed by the introduction of new :: constructors), which gives reverse a complexity of $O(n^2)$. This is indicated by the sketch to the right of the above trace.

[edit] Tail-Recursive Implementation

By introducing an accumulator, we can write list reversal tail-recursively as follows:

```
(*
    val reverse': 'a list -> 'a list
        reverse' l reverses the order of the elements in the list l
    Invariants: none
    Effects:    none
*)
fun reverse' (l: 'a list): 'a list =
    let
        (*
        val rev: 'a list * 'a list -> 'a list
            rev(l, acc) ==> <reverse of list l> @ acc
        Invariants: none
        Effects:    none
        *)
        fun rev(nil: 'a list, acc: 'a list): 'a list = acc
```

# Wiki-Based Instruction

➢ Not just a surrogate for a book!
➢ A comprehensive didactic tool
  ▪ Promotes participatory learning
    • Students are "encouraged" to modify articles
      – For play, for curiosity, or for points
      – Get them to research topics
      – Explain ideas to others (in writing)
    • Active participants in the didactic process
  ▪ Easy monitoring of students' involvement
    • Every edit is logged
      – We know who did what
    • Every access is logged
      – Make sure students read the material before class
        » More interactive and focused in-class discussion
        » No "guest account"

# Actual Experiments

➢ 1 recitation on wiki editing

➢ 20% of 1st assignment

- Objective 1: play with the mechanics of the wiki
- Objective 2: test research/creativity

Problem 4: The 15-212 Wiki [10 Points] Due by Tuesday, January 22nd

This part (and only this part) will be done with a partner. At recitation 2 on January Sunday 20th, you will be introduced to the 15-212 wiki. As an assignment, you will have to write a wiki page on a given topic. One of these topics will be assigned to you:

- **Strings and Boolean:** You work will be to write something about String and Boolean in the same way that it has been already done for Integers and Reals.

- **The Standard ML Basis Libraries:** You work will be to introduce what are the ML basis libraries and briefly describe the most useful ML libraries.

- **SML Top Level Environment:** You work will be to explain what is natively defined in the SML Top Level Environment.

- **ML compilers:** Your work will be to write a short summary about the other existing ML compilers and their differences with SML/NJ.

8 students

Group A

Group B

Group C

Group D

# Group A

- Students followed models rather accurately
  - Took good advantage of wiki
  - Good starting point

- Some humor

- Little creativity
  - Shy to experiment

Carnegie Mellon
QATAR CAMPUS
15-212 WIKI

## Strings (intro version)

... for some lucky students to fill ...

["UPDATE"] WE WERE LUCKY TO DO SO!!!

**Contents** [hide]
1 Syntax
  1.1 Type
  1.2 Values
  1.3 Expression
2 Semantics
  2.1 Typing Rules
  2.2 Evaluation Rules
  2.3 Examples
3 Advanced Material
4 External References

### [edit] Syntax

The syntax for string types is specified in the STRING signature as part of the SML Basis Library STRING. The most common structure for the boolean is the String structure.

### [edit] Type

A string is a sequence of characters. The SML representation of the string is classified by the type

**string**.

### [edit] Values

A string can take on any of the possible given values:
- Letters
- Numbers
- Symbols
- and other miscellaneous characters.

There is no pre-defined order of writing them.

**N.B**: To print out characters such as " or new line, they have to be pre-pennded by a backslash(\) which makes them an *escape sequence* e.g.

```
"1234" is printed as 1234, while "\"1234\"" is printed as "1234".
```

### [edit] Expression

Strings have the flexibility of being merged together, which is known as *concatenation*. It can be performed as follows:

- $e_1 \char`^ e_2$ **iff** $e_1$:string & $e_2$:string

### [edit] Semantics

- *val it = "1234":string*

### [edit] Typing Rules

- Every string value has type **string**.

  $always$   x : **string**

- Every one of the String expressions expects to have a type **string** as arguments. So, when typechecking some expression $e_1 \char`^ e_2$ the ML typechecker takes the following steps:

  1. It checks that $e_1$ has type **string** (Otherwise it will returns an error)

# Group A (2)

➢ Same structure
  ▪ Was partially given

➢ Lot less prose
  ▪ Ran out of time?
    • 2nd week of class

# Group B

➢ These students looked up the material and reported on what they found

- No elaboration
- Not integrated within wiki
- Limited use to other students

➢ Could have written more

---

article | discussion | edit | history | protect | delete | move | watch

## SML/NJ Basis Library

**Contents** [hide]
1 What is the SML/NJ Basis Library ?
2 How can I access the library ?
3 Frequently used library references
   3.1 Top-level Environment
   3.2 Basic Types
   3.3 Standard Datatypes
   3.4 Arrays and Vectors
   3.5 String
4 SML Basis Manual

### [edit] What is the SML/NJ Basis Library ?

The SML/NJ Basis Library is like API(Application Programming Interface) specifications for the core modules available in the SML programming language. It provides interfaces and operations for basic types, such as integers and strings, support for input and output (I/O), interfaces to basic operating system interfaces, and support for standard datatypes, such as options and lists. However the library does not contain the higher level API and information about GUI related libraries.

### [edit] How can I access the library ?

Follow this 🔗 link or, copy paste the following link in your browser:

http://www.standardml.org/Basis/

### [edit] Frequently used library references

### [edit] Top-level Environment

In order to get a brief understanding of Top-level Environment go to SML/NJ top level environment. To access it in the SML/NJ Basis Library, follow SML Top-Level Environment 🔗

### [edit] Basic Types

int, real, bool, word, Math

### [edit] Standard Datatypes

option 🔗, list 🔗

### [edit] Arrays and Vectors

array 🔗, vector 🔗

### [edit] String

string 🔗, char 🔗, substring 🔗

### [edit] SML Basis Manual

Follow this 🔗 link or, copy paste the following link in your browser:

http://www.standardml.org/Basis/manpages.html

Category: ML

# Group C

- ➢ Students did research and found several relevant documents
  - ▪ Good analysis/synthesis work
  - ▪ Combine prose and code as appropriate
  - ▪ Reference sources

- ➢ Lots of humor and creativity
  - ▪ Text at the top of the page
  - ▪ Cartoon

- ➢ Best result

## SML/NJ top level environment

So! You got the SML/NJ prompt. What symbols have been defined and are available? What can one do? How does one get out?

The *top-level environment* consists of those identifiers that are available upon login, before the user introduces additional top-level bindings. There are two reasons for including (non-module) identifiers in the top-level environment. The first is convenience. Certain types and values are used so frequently that it would be perverse to force the programmer to always open the containing structures or to use the qualified names. This is particularly true for interactive interfaces, in which notational simplicity and fewer keystrokes are desirable. The second reason is to allow operator overloading.

**Contents** [hide]
1 Infix Identifiers
2 Top-Level Types
3 Exception Constructors
4 Overload Identifiers
5 Using the Environment
6 Exiting the Environment
7 References

## [edit] Infix Identifiers

The **Infix Identifiers** (that appear between two operands) in SML are:

```
infix  7   * / div mod
infix  6   + - ^
infix  5   :: @
infix  4   = <> > >= < <=
infix  3   := o
infix  0   before
```

## [edit] Top-Level Types

The following **Top-Level Types** (and their defining structures, if any) occur in SML:

```
eqtype unit                              General
eqtype int                               Int
eqtype word                              Word
type real                                Real
eqtype char                              Char
eqtype string                            String
type substrig                            Substrin
type exn                                 General
eqtype 'a array                          Array
eqtype 'a vector                         Vector
eqtype 'a ref                            primiti
datatype bool = false | true             primiti
datatype 'a option = NONE | SOME of 'a   Option
datatype order = LESS | EQUAL | GREATER  General
datatype 'a list = nil | :: of ('a * 'a list)  primiti
```

## [edit] Exception Constructors

The following **Exception Constructors** are available in SML:

```
exception Bind
exception Chr
exception Div
```
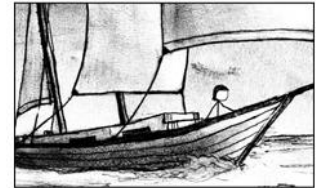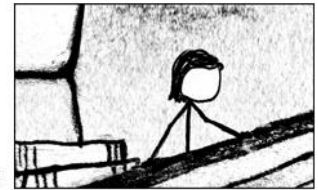
# Group D

> Students did a search on the web and dumped what they found
- Little post-processing
- No creativity, no fun

> I was not too happy about this one

## ML compilers

### [edit] Standard ML of New Jersey

Also known as SML/NJ, it is an optimizing native-code compiler for Standard ML that is written in Standard ML. It runs on a wide range of architectures. It was developed jointly by Bell Laboratories and Princeton University.

### [edit] Moscow ML

Moscow SML is particularly suitable for teaching and experimentation, where fast compilation and modest storage consumption are more important than fast program execution. Thanks to the efficient run-time system of Caml Light, Moscow SML compiles fast and uses little memory. Moscow ML was created by Sergei Romanenko at the Keldysh Institute of Applied Mathematics, Russian Academy of Sciences, Moscow, Claudio Russo at Cambridge University, UK, Niels Kokholm at the IT University of Copenhagen (Moscow ML for .Net), Ken Friis Larsen at the IT University of Copenhagen, and Peter Sestoft at the Royal Veterinary and Agricultural University, Copenhagen, Denmark.

### [edit] MLton

MLton aims to produce fast executables, and to encourage rapid prototyping and modular programming by eliminating performance penalties often associated with the use of high-level language features. It was developed by Suresh Jagannathan, Stephen Weeks, Matthew Fluet and Henry Cejtin, who is also the first employee of Mathematica.

### [edit] Poly/ML

Poly/ML was originally written by David Matthews at the Computer Laboratory at Cambridge University.

### [edit] TILT

TILT is a self-checking compiler for Standard ML that uses Typed Intermediate Languages. TILTis under active development as part of the ConCert Project at Carnegie Mellon. TILT began as a joint effort between researchers at Cornell and in the Fox Project at Carnegie Mellon to develop a successor to the TIL Compiler, a self-checking compiler for the Standard ML core language.

### [edit] The ML Kit

The ML Kit compiler covers all of Standard ML, as defined in the 1997 edition of the Definition of Standard ML. The MLKit implements most of the latest Standard ML Basis Library specification. The people involved in The ML Kit can be found on this page.

### [edit] SML.NET

SML.NET is a compiler for the functional programming language Standard ML that targets the .NET Common Language Runtime and which supports language interoperability features for easy access to .NET libraries. The people involved in SML.NET can be found on this page.

Category: ML

# Outcome Summary

- Objective 1: play with mechanics of wiki
  - All figured out the basics
  - Some did a little extra
  - None did more than expected
- Objective 2: test research/creativity
  - Research varied from vigorous web search to minimum needed to adapting example
  - Creativity ranged from the dull to the surprising
- Altogether
  - Students found exercise fun
    - Novelty
    - Not usual routine

# The Wiki and 212's Future

- Categorization helps focus on the big picture
  - What the course is about
    - Problem solving, not ML
- Encourages rearranging material and delivery
  - Experiment with what works best
    - Case-studies, examples, exercises
    - Interplay between problem solving and programming
- Encourage exploring syllabus improvements
  - Add/remove topics
  - Change language

- A more dynamic course

# Future Developments

- Add rest of course material
  - … my summer project
  - Continuous improvement cycle
- Extend with "try-it" capability
  - Sandboxed interpreter within wiki
- Pair-up exercises with e-tutor
  - Intelligent learning system
- Explore opportunistic learning …

# Opportunistic Learning

- Books, notes often modeled after lecture
  - 1+ hour long
    - Long attention span
  - Mostly linear presentation of material
  - This is not necessary and probably not efficient
- Wiki breaks away from this model
  - Brief topic-oriented articles linked together
    - Smaller time granularity for studying/reviewing (5-20 min)
      - Harness "dead times" (commuting, time between classes, …)
    - Focus on actual dependencies
  - Make it mobile
    - Reformat wiki for viewing on PDAs, smartphones
    - Add matching video segments of lecture, slides, …