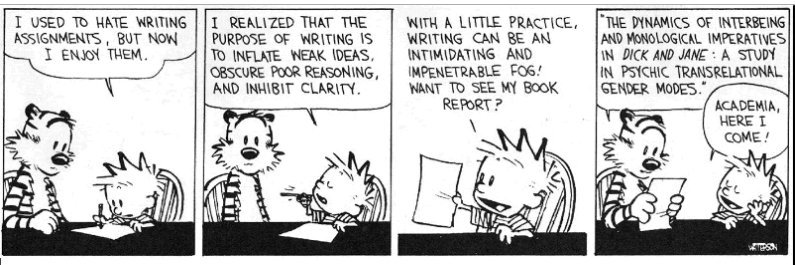


Languages

An-cay uo-yay eak-spay
ig-pay atin-lay?

#1

Problem Set 0

- **PS 0 Due Tomorrow at Midnight**
  - Anyone can turn it in (waitlist = OK)
- Structured Lab Hour Today, OLS 001, 7-8, 8-9
- The answers are not long
  - Some have already completed it
  - Reading the problem may take the longest
- Two standard approaches to CS problems:
  - "Flail around randomly" until it works (long!)
  - "Think very hard; write down the answer" (short!)
    - Apologies to Feynman and Gell-Mann

#2

Outline

- Languages and Formal Systems
- BNF Grammars
- Describing Languages
- Learning New Languages
- Evaluation Rules

#3

What is a language?

Webster:

A systematic means of communicating ideas or feelings by the use of conventionalized signs, sounds, gestures, or marks having understood meanings.

#4

What is a language?

Webster:

A ~~systematic~~ means of communicating ~~ideas or feelings~~ by the use of ~~conventionalized~~ signs, sounds, gestures, or marks having ~~understood~~ meanings.

#5

Linguist's Definition

(Charles Yang)

A **language** is:

A description of pairs (*S*, *M*), where *S* stands for sound, or any kind of surface forms, and *M* stands for meaning.

A theory of language must specify the properties of *S* and *M*, and how they are related.

#6

Languages and Formal Systems

What is the difference between a formal system and a language?

With a language, the surface forms have **meaning**.

Caveat: computer scientists often use *language* to mean just a set of surface forms.

What are languages made of?

- **Primitives** (almost all languages have these)
  - The simplest surface forms with **meaning**
- **Means of Combination** (**all** languages have these)
  - Like Rules of Production for Formal Systems
  - Ways to make new surface forms from ones you already have
- **Means of Abstraction** (all **powerful** languages have these)
  - Ways to use simple surface forms to represent complicated ones

Does English have these?

- Primitives
  - Words (?)
- Means of combination
  - ?

Does English have these?

- Primitives
  - ~~Words (?)~~
    - e.g., "antifloccipoccinihilipilification" - **not** a primitive
  - Morphemes – smallest units of meaning
    - e.g., **anti-** ("opposite")
- Means of combination
  - e.g., *Sentence* ::= *Subject Verb Object*
  - Precise rules, but not the ones you learned in grammar school

Ending a sentence with a preposition is something up with which we will not put.
Winston Churchill

Does English have these?

- Means of abstraction
  - Pronouns: she, he, it, they, which, etc.
  - Confusing since they don't always mean the same thing, it depends on where they are used.

The "**these**" in the slide title is an abstraction for the three elements of language introduced 2 slides ago.

The "**they**" in the confusing sentence is an abstraction for pronouns.

How should we describe (Formal) Languages?

Backus Naur Form

symbol ::= replacement

We can replace *symbol* with *replacement*

> *A* ::= *B* means anywhere you have an *A*, you can replace it with a *B*.

nonterminal - symbol that appears on left side of rule

terminals - symbol that **never** appears on the left side of a rule

BNF Example

Sentence ::= *NP Verb*
NP ::= *Noun*
Noun ::= **Wes**
Noun ::= **Scheme**
Verb ::= **rocks**
Verb ::= **sucks**

What are the *terminals*?

How many different things can we express with this language?

BNF Example

Sentence ::= *NP Verb*
NP ::= *Noun*
Noun ::= **Wes**
Noun ::= **Scheme**
Verb ::= **rocks**
Verb ::= **sucks**

What are the *terminals*?

Wes, **Scheme**, **rocks**, **sucks**

How many different things can we express with this language?

4, but only 2 are true.

BNF Example

Sentence ::= *NP Verb*
NP ::= *Noun*
NP ::= *Noun* **and** *NP*
Noun ::= **Wes**
Noun ::= **Scheme**
Verb ::= **rocks**
Verb ::= **sucks**

How many different things can we express with this language?

BNF Example

Sentence ::= *NP Verb*
NP ::= *Noun*
NP ::= *Noun* **and** *NP*
Noun ::= **Wes**
Noun ::= **Scheme**
Verb ::= **rocks**
Verb ::= **sucks**

How many different things can we express with this language?

Infinitely many!
Recursion is powerful.

Liberal Arts Trivia: Art History

- Q. Name the type of painting in which pigment is mixed with water on a thin layer of mortar or plaster. Because of the chemical makeup of the plaster, a binder is not required, as the pigment mixed solely with the water will sink into the intonaco, which itself becomes the medium holding the pigment. The technique was popular during the European Renaissance.

Liberal Arts Trivia: Music

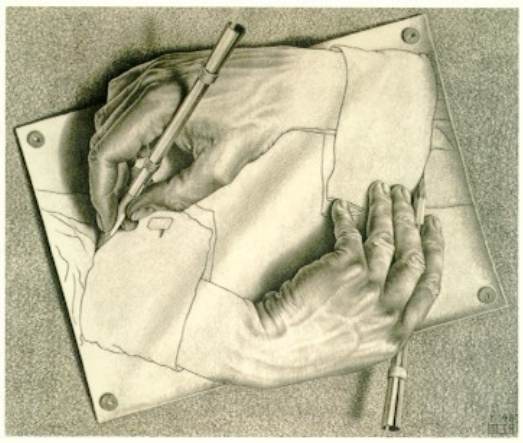
- Q. This Hong Kong singer is one of the original four cantopop Heavenly Kings (四大天王), and possesses a rich baritone/tenor. He is sometimes called the God of Songs (歌神). His most famous work is perhaps Goodbye Kiss (吻別) – one of the best-selling albums of all time, with over 3 million copies sold in 1993 alone. Give the English or Romanized name of this singer.

Most Essential Scheme

Expr	::= *PrimitiveExpr*
PrimitiveExpr	::= ***Number***
PrimitiveExpr	::= **+** | ***** | **<=** | ...
Expr	::= ***Name***
Expr	::= *ApplicationExpr*
ApplicationExpr	::= **(***Expr MoreExprs***)**
MoreExprs	::=
MoreExprs	::= *Expr MoreExprs*

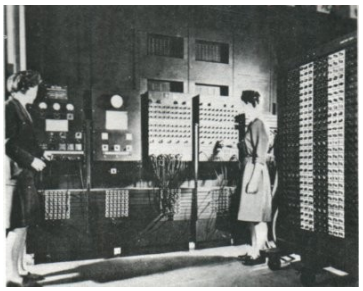
This is everything you need to write for PS0 and PS1 !

Rules of Evaluation & People



ENIAC: Electronic Numerical Integrator and Computer

- Early WWII computer
  - But **not** the world's first (PS4)
- Built to calculate bombing tables



Memory size:
twenty 10 decimal digit accumulators = 664 bits

ENIAC (1946): 3 mm
Apollo Guidance Computer (1969): 1 inch
You: 2.2 miles

Directions for Getting **6**

1. Choose any regular accumulator (ie. Accumulator #9).
2. Direct the Initiating Pulse to terminal *5i*.
3. The initiating pulse is produced by the initiating unit's *Io* terminal each time the Eniac is started. This terminal is usually, by default, plugged into Program Line 1-1 (described later). Simply connect a program cable from Program Line 1-1 to terminal *5i* on this Accumulator.
4. Set the Repeat Switch for Program Control 5 to 6.
5. Set the Operation Switch for Program Control 5 to ADD.
6. Set the Clear-Correct switch to C.
7. Turn on and clear the Eniac.
8. Normally, when the Eniac is first started, a clearing process is begun. If the Eniac had been previously started, or if there are random neons illuminated in the accumulators, the ``Initial Clear'' button of the Initiating device can be pressed.
9. Press the ``Initiating Pulse Switch'' that is located on the Initiating device.
10. **Stand back.**

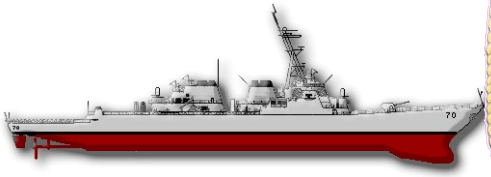
Admiral Grace Hopper

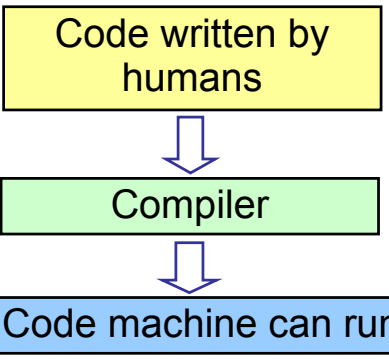
(1906-1992)

"Nobody believed that I had a running compiler and nobody would touch it. They told me computers could only do arithmetic."

- Mathematics PhD Yale, 1934
- Entered Navy, 1943
- First to program Mark I (first "large" computer, 51 feet long)
- Wrote first compiler (1952) – program for programming computers
- Co-designer of COBOL (most widely used programming language until a few years ago)

USS Hopper



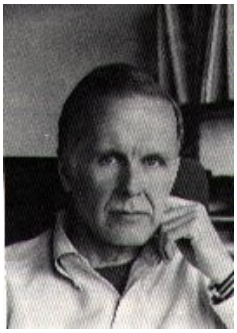


Compiler translates from code in a high-level language to machine code

DrScheme uses an ***interpreter***. An interpreter is like a compiler, except it runs quickly and quietly on small bits of code at a time.

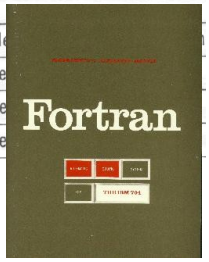
John Backus

- Chemistry major at UVA (entered 1943)
- Flunked out after second semester
- Joined IBM as programmer in 1950
- Developed Fortran, first commercially successful programming language and compiler



IBM 704 Fortran manual, 1956

STATEMENT	NORMAL SEQUENCING
a = b	Next executable statement
GO TO n	Statement n
GO TO n, $(n_1,n_2,\ldots,n_m)$	Statement last assigned
ASSIGN i TO n	Next executable statement
GO TO $(n_1,n_2,\ldots,n_m)$, i	Statement n_i
IF (a) n_1,n_2,n_3	Statement n_1,n_2,n_3 as a l
SENSE LIGHT i	Next executable stateme
IF (SENSE LIGHT i) n_1,n_2	Statement n_1,n_2 as Sense
IF (SENSE SWITCH i) n_1,n_2	" " " as Sense



Describing Languages

- Fortran language was described using English
  - Imprecise
  - Verbose, lots to read
  - Ad hoc

```
DO 10 I=1.10
```

Assigns `1.10` to the variable `DO10I`

```
DO 10 I=1,10
```

Loops for `I = 1` to `10`

(Often incorrectly blamed for loss of Mariner-I)

- Wanted a more precise way of describing a language

Recall: Backus Naur Form

symbol ::= *replacement*

We can replace *symbol* with *replacement*

A ::= *B* means anywhere you have an *A*, you can replace it with a *B*.

nonterminal - symbol that appears on left side of rule

*terminal*s - symbol that **never** appears on the left side of a rule

Language Elements

When learning a foreign language, which elements are hardest to learn?

- Primitives: lots of them, and hard to learn real *meaning*
- Means of Combination
  - Complex, but, all natural languages have similar ones [Chomsky]

SOV (45% of all languages)	*Sentence* ::= *Subject Object Verb*	(Korean)
SVO (42%)	*Sentence* ::= *Subject Verb Object*	
VSO (9%)	*Sentence* ::= *Verb Subject Object*	(Welsh)
	"Lladdodd y ddraig y dyn."	
	(Killed the dragon the man.)	
OSV (<1%):	Tobati (New Guinea)	
Schemish:	*Expression* ::= (*Verb Object*)	

- Means of Abstraction: few of these, but tricky to learn differences across languages

  English: I, we

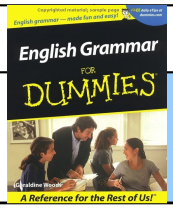
  Tok Pisin (Papua New Guinea): mi (I), mitupela (he/she and I), mitripela (both of them and I), mipela (all of them and I), yumitupela (you and I), yumitripela (both of you and I), yumipela (all of you and I)

	Pages in *Revised*[5] *Report on the Algorithmic Language Scheme*
Primitives	
Means of Combination	
Means of Abstraction	
	48 pages total (includes formal specification and examples)

	Pages in *Revised*[5] *Report on the Algorithmic Language Scheme*		
Primitives	Standard Procedures	18	
	Primitive expressions	2	
	Identifiers, numerals	1	
Means of Combination	Expressions	2	
	Program structure	2	
Means of Abstraction	Definitions	½	
	48 pages total (includes formal specification and examples)		

	Pages in *Revised*[5] *Report on the Algorithmic Language Scheme*		Pages in C++ Language Specification (1998)
Primitives	Standard Procedures	18	356
	Primitive expressions	2	30
	Identifiers, numerals	1	10
Means of Combination	Expressions	2	**C++ Core language issues list has 469 items!** 197
	Program structure	2	35
Means of Abstraction	Definitions	½	Declarations, Classes 173
	48 pages total (includes formal specification and examples)		**776** pages total (includes no formal specification or examples)

	Pages in *Revised*[5] *Report on the Algorithmic Language Scheme*		English
Primitives	Standard Procedures	18	Morphemes ?
	Primitive expressions	2	Words in Oxford English Dictionary 500,000
	Identifiers, numerals	1	
Means of Combination	Expressions	2	Grammar Rules 100s (?)
	Program structure	2	*English Grammar for Dummies* Book 384 pages
Means of Abstraction	Definitions	½	Pronouns ~20
	48 pages total (includes formal specification and examples)		

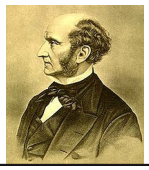


Liberal Arts Trivia: Architecture

- Q. Name the distinctive architectural features of Islamic mosques that is typically a spiresor onion-shaped dome, and is usually either free-standing or much taller than all surrounding structures. There are six in this picture of the Sultan Ahmed Mosque (Blue Mosque) in Istanbul:

Liberal Arts Trivia: Philosophy

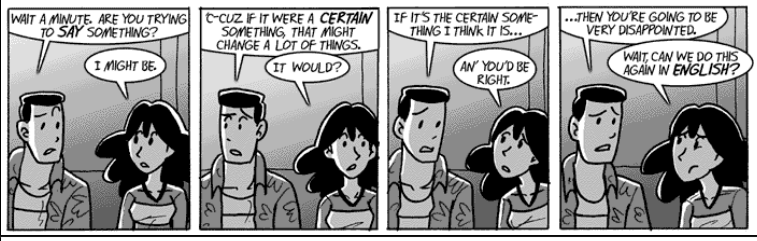
- Q. Name this 19th century philosophical work by John Stuart Mill. To the Victorian readers of the time it was radical, advocating moral and economic freedom of individuals from the state. Mill argues against the "tyranny of the majority" and articulates the harm principle: people can do anything they like as long as it does not harm others.



Evaluation Rules - Step by Step

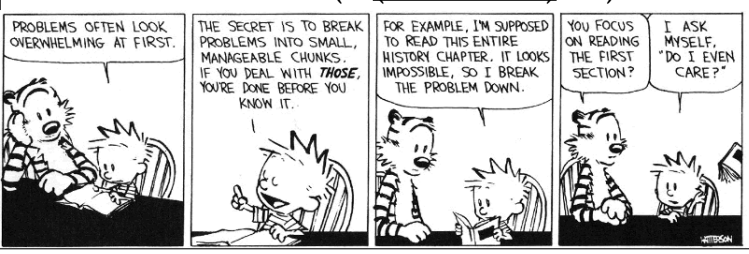
Expressions and Values

- (Almost) every ***expression*** has a ***value***
  - Have you seen any expressions that don't have values?
- When an expression with a value is ***evaluated***, its value is produced



Five Types of Expression

1. **Primitives** (if #t 3 5)
2. **Names** (define (square n) (* n n))
3. **Application** (square 4)
4. **Lambda** ((lambda (q) (- 0 q)) 7)
5. **If** (+ (if true 3 5) 10)



Primitive Expressions

Expression ::= *PrimitiveExpression*

PrimitiveExpression ::= ***Number***
PrimitiveExpression ::= **#t** | **#f**
PrimitiveExpression ::= ***Primitive Procedure***

Evaluation Rule 1: Primitives

If the expression is a ***primitive***, it evaluates to its pre-defined value.

```
> 2
2
> #t
#t
> +
#<primitive:+>
```


Name Expressions

Expression ::= *NameExpression*

NameExpression ::= ***Name***

Evaluation Rule 2: Names

If the expression is a ***name***, it evaluates to the value associated with that name.

```
> (define two 2)
> two
2
```


Application Expressions

Expression ::= *Application Expression*

ApplicationExpression
::= (*Expression MoreExpressions*)

MoreExpressions ::= ε

MoreExpressions
::= *Expression MoreExpressions*

Evaluation Rule 3: Application

3. If the expression is an application:
   a) **Evaluate** all the subexpressions (in any order)
   b) **Apply** the value of the first subexpression to the values of all the other subexpressions.

(Expression$_0$ Expression$_1$ Expression$_2$ …)

Rules for Application

I. **Primitives.** If the procedure to apply is a *primitive*, just do it.

II. **Constructed Procedures.** If the procedure is a *constructed procedure*, **evaluate** the body of the procedure with each formal parameter replaced by the corresponding actual argument expression value.

Eval and Apply are **defined in terms of each other**.

Without Eval, there would be no Apply, Without Apply there would be no Eval!

Making Procedures

lambda means "make a procedure"

Expression ::= *ProcedureExpression*

ProcedureExpression ::=
(**lambda** (*Parameters*) *Expression*)

Parameters ::= ε

Parameters ::= *Name Parameters*

Evaluation Rule 4: Lambda

4. Lambda expressions evaluate to a procedure that takes the given parameters and has the expression as its body.

Lambda is the English name for the Greek letter written λ .

Lambda Example: Tautology Function

```
(lambda      make a procedure
 ()          with no parameters
 #t)         with body #t

> ((lambda () #t) 150)
#<procedure>: expects no arguments, given 1: 150
> ((lambda () #t))
#t
> ((lambda (x) x) 150)
150
```


Evaluation Rule 5: If

(**if** $Expression_{Predicate}$
$Expression_{Consequent}$
$Expression_{Alternate}$)

To evaluate an if expression:

(a) Evaluate $Expression_{Predicate}$.

(b) **If** it evaluates to #f, the value of the if expression is the value of $Expression_{Alternate}$. **Otherwise**, the value of the if expression is the value of $Expression_{Consequent}$.

Now You Know All of Scheme!

- Once you understand Eval and Apply, you can understand all Scheme programs!
- Except:
  - There are a few more special forms (like **if**)
  - We have not define the evaluation rules precisely enough to unambiguously understand all programs (e.g., what does "value associated with a name" mean?)

Example: Nanostick

- How far does light travel in 1 nanosecond?

```
> (define nanosecond (/ 1 (* 1000 1000 1000))) ;; 1 billionth of a s
> (define lightspeed 299792458)                 ; m / s
> (* lightspeed nanosecond)
149896229/500000000
> (exact->inexact (* lightspeed nanosecond))
0.299792458
```

= just under 1 foot

Dell machines in Small Hall have “1.8-GHz Pentium 4 CPU”

GHz = GigaHertz = 1 Billion times per second
They must finish a step before light travels 6.6 inches!

Homework

- Read Structured Lab Guide
  - Before Structured Lab Hours (Today 7-8 or 8-9)
- Problem Set 0 due Thursday at Midnight
- No Class Monday (MLK Day)
- Read GEB pp. 3-41 by next Wednesday