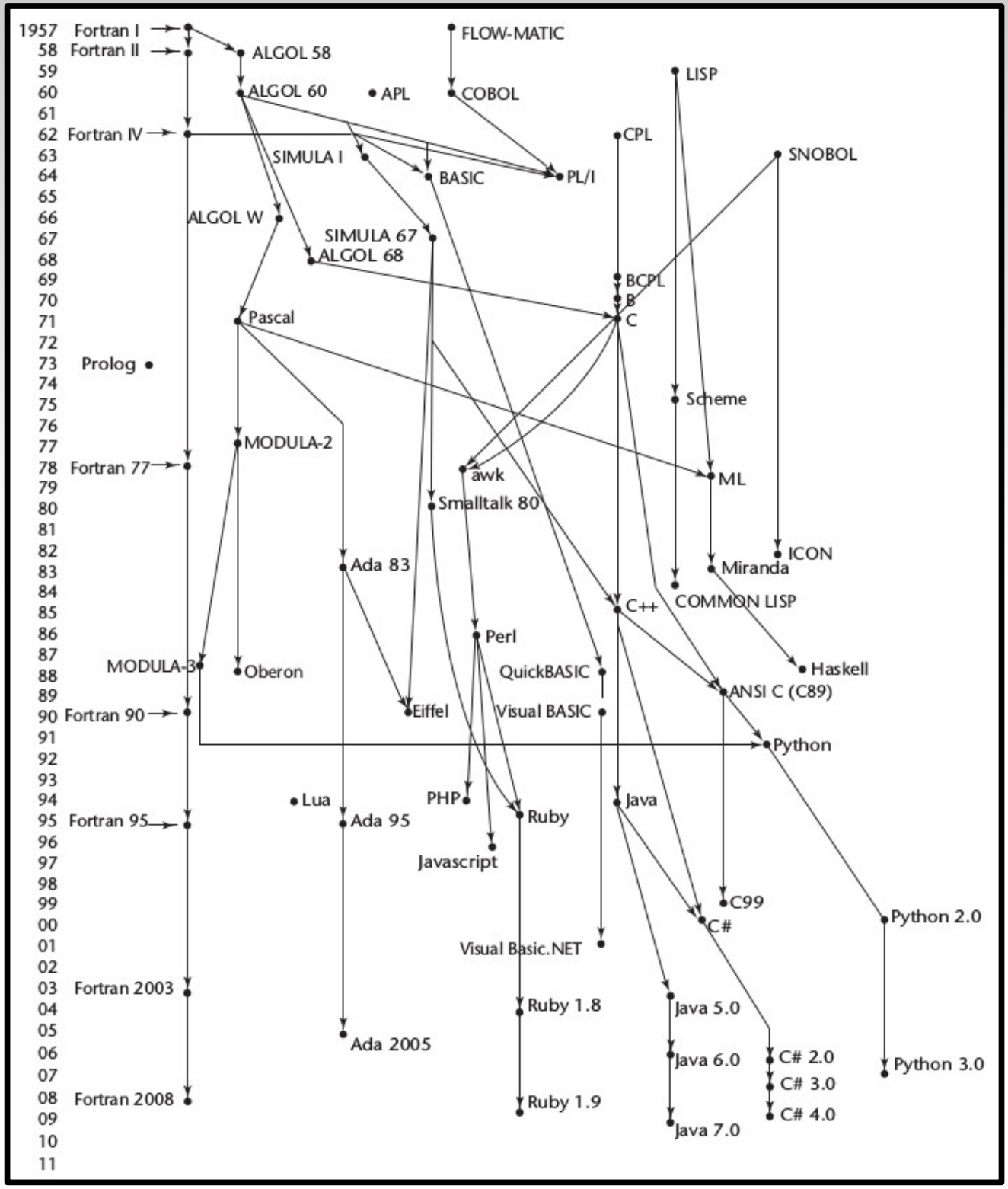
Linguagens de Alto Nível

Estrutura de Linguagens

https://github.com/fsantanna-uerj/EDL/

Francisco Sant'Anna

francisco@ime.uerj.br



Linguagens de "Alto Nível"

```
MODE DATA = REF CHAR;

PROC in place insertion sort = (REF[]DATA item)VOID:
BEGIN
   INT first := LWB item;
   INT last  := UPB item;
   INT j;
   DATA value;
   FOR i FROM first + 1 TO last DO
      value := item[i];
      j := i - 1;
   #  WHILE j >= LWB item AND j <= UPB item ANDF item[j] > value DO // ex
      WHILE ( j >= LWB item AND j <= UPB item | item[j]>value | FALSE ) D
         item[j + 1] := item[j];
         j -:=  1
      OD;
      item[j + 1] := value
   OD
END # in place insertion sort #;

[32]CHAR data := "big fjords vex quick waltz nymph";
[UPB data]DATA ref data;  FOR i TO UPB data DO ref data[i] := data[i] OD;
in place insertion sort(ref data);
FOR i TO UPB ref data DO print(ref data[i]) OD; print(new line);
print((data))
```

ALGOL

```
C-PROCESS SECTION.
    PERFORM E-INSERTION VARYING WB-IX-1 FROM 1 BY 1
                        UNTIL WB-IX-1 > WC-SIZE.



E-INSERTION SECTION.
E-000.
    MOVE WB-ENTRY(WB-IX-1) TO WC-TEMP.
    SET WB-IX-2 TO WB-IX-1.

    PERFORM F-PASS UNTIL WB-IX-2 NOT > 1 OR
                         WC-TEMP NOT < WB-ENTRY(WB-IX-2 - 1).

    IF WB-IX-1 NOT = WB-IX-2
       MOVE WC-TEMP TO WB-ENTRY(WB-IX-2).

E-999.
    EXIT.

F-PASS SECTION.
F-000.
    MOVE WB-ENTRY(WB-IX-2 - 1) TO WB-ENTRY(WB-IX-2).
    SET WB-IX-2                DOWN BY 1.

F-999.
    EXIT.
```

COBOL

```
      SUBROUTINE SORT(N,A)
      IMPLICIT NONE
      INTEGER N,I,J
      DOUBLE PRECISION A(N),X
      DO 30 I = 2,N
        X = A(I)
        J = I
10      J = J - 1
        IF (J.EQ.0) GO TO 20
        IF (A(J).LE.X) GO TO 20
        A(J + 1) = A(J)
        GO TO 10
20      A(J + 1) = X
30    CONTINUE
      END
```

Fortran

```
(defun span (predicate list)
  (let ((tail (member-if-not predicate list)))
    (values (ldiff list tail) tail)))

(defun less-than (x)
  (lambda (y) (< y x)))

(defun insert (list elt)
  (multiple-value-bind (left right) (span (less-than elt) list)
    (append left (list elt) right)))

(defun insertion-sort (list)
  (reduce #'insert list :initial-value nil))
```

LISP

Exercícios

1. Quais linguagens nos slides que você…
   - já tinha ouvido falar? em que situação?
   - já teve algum contato? em que nível?
   - programa com frequência? em que contexto?
   - mais teria curiosidade de conhecer? por quê?
2. Dentre as linguagens que você já usou, …
   - qual você mais gosta? por quê?
   - qual você menos gosta? por quê?

Linguagens de Alto Nível

(continuação...)

Estrutura de Linguagens

Francisco Sant'Anna

Sala 6020-B
francisco@ime.uerj.br
http://github.com/fsantanna-uerj/EDL

Linguagens de Baixo Nível

- Código de Máquina

```
8B542408 83FA0077 06B80000 0000C383
FA027706 B8010000 00C353BB 01000000
C9010000 008D0419 83FA0376 078BD98B
B84AEBF1 5BC3
```

- Mapeamento 1:1 para CPU
  - Máquina imperativa com espaço de endereçamento plano
- Binário vs Assembly
  - Mnemônicos, Offsets, Endereços Simbólicos
- Não estamos interessados nelas
  - São consequência direta da CPU

- Assembly

```
    mov edx, [esp+8]
    cmp edx, 0
    ja @f
    mov eax, 0
    ret

    @@:
    cmp edx, 2
    ja @f
    mov eax, 1
    ret

    @@:
    push ebx
    mov ebx, 1
    mov ecx, 1

    @@:
        lea eax, [ebx+ecx]
        cmp edx, 3
        jbe @f
        mov ebx, ecx
        mov ecx, eax
        dec edx
    jmp @b

    @@:
    pop ebx
    ret
```

Linguagens de Alto Nível

- Portabilidade
  - detalhes de arquitetura (registradores, alinhamento)
  - sintaxe uniforme
- Produtividade
  - abstrações de dados (tipos, registros, vetores, classes)
  - abstrações de controle (loops, rotinas, continuações)
  - concorrência, domínio, etc
- Performance?
  - otimizações globais
  - instruções específicas
  - complexidade

```
unsigned int [redacted](unsigned int n)
{
    if (n <= 0)
        return 0;
    else if (n <= 2)
        return 1;
    else {
        unsigned int a,b,c;
        a = 1;
        b = 1;
        while (1) {
            c = a + b;
            if (n <= 3) return c;
            a = b;
            b = c;
            n--;
        }
    }
}
```

Exercícios

- Escolha uma linguagem de sua preferência…
  1. Sobre portabilidade...
     - em que sistemas operacionais ela está disponível?
     - em que arquiteturas ela está disponível?
     - em que sistema ou arquitetura ela **não** está disponível?
  2. Sobre abstações de dados…
     - que abstrações diferentes/especiais ela possui?
     - que abstrações **comuns** ela não possui?
  3. Sobre abstrações de controle…
     - que abstrações diferentes/especiais ela possui?
     - que abstrações **comuns** ela não possui?
  4. Sobre uma abstração de domínio específico
     - como ela ajuda na produtividade?

Linguagens de Alto Nível

(continuação...)

Estrutura de Linguagens

Francisco Sant'Anna

Sala 6020-B
francisco@ime.uerj.br
http://github.com/fsantanna-uerj/EDL

Linguagem de Programação

```c
unsigned int fib(unsigned int n)
{
    if (n <= 0)
        return 0;
    else if (n <= 2)
        return 1;
    else {
        unsigned int a,b,c;
        a = 1;
        b = 1;
        while (1) {
            c = a + b;
            if (n <= 3) return c;
            a = b;
            b = c;
            n--;
        }
    }
}
```

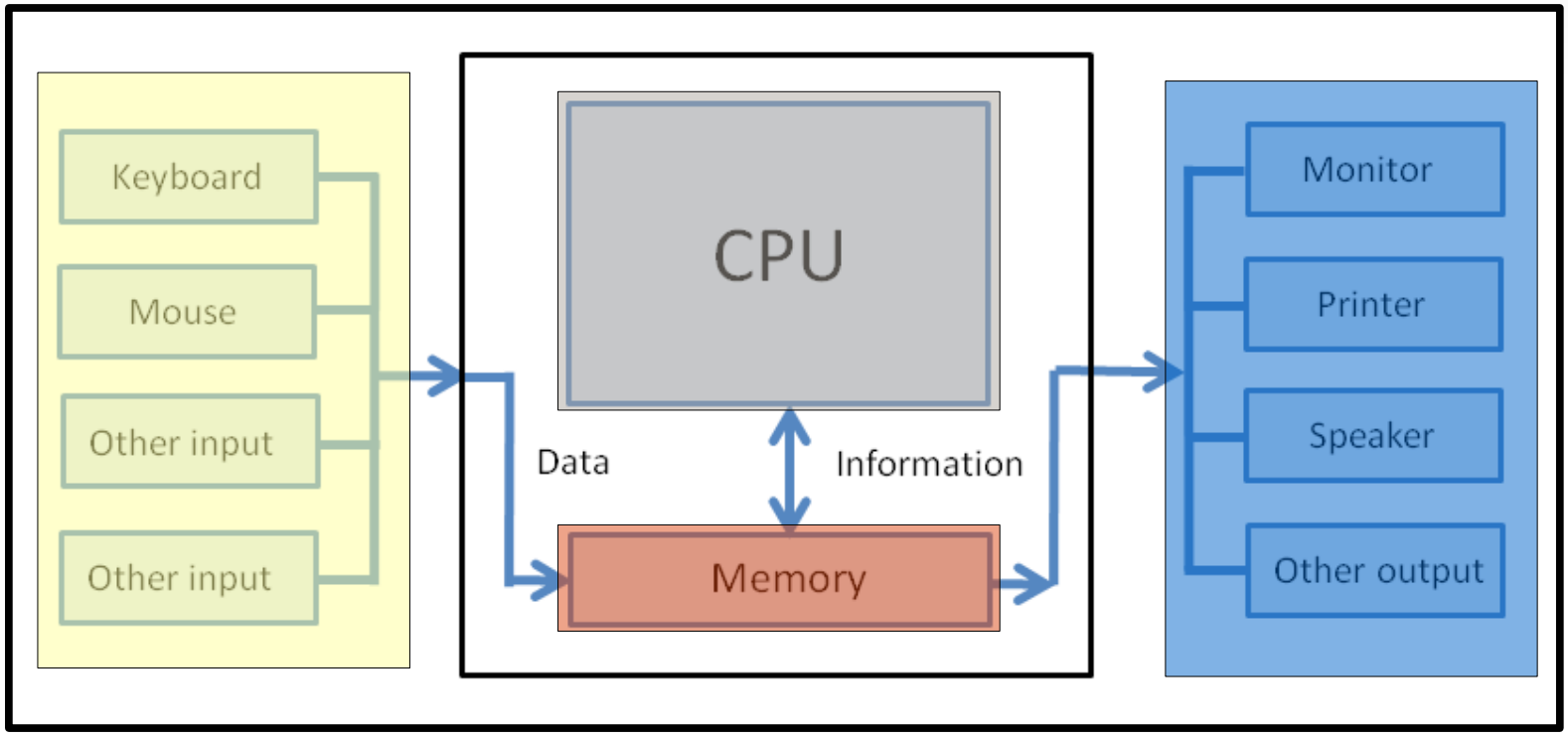
Compilador de C

```
8B542408 83FA0077 06B80000 0000C383
FA027706 B8010000 00C353BB 01000000
C9010000 008D0419 83FA0376 078BD98B
B84AEBF1 5BC3
```

as

Linguagem como Abstração

```
frase = input()
print("----")
for i in range(1,5):
    print(i, frase)
```



Sintaxe vs Semântica

- *Forma*, *Símbolos* vs Significado, Execução
- Exemplo: Como é o comando **`while`** de C?
  - *Sintaxe:*
    - `While ::= while ( Expression ) Statement`
    - Formal, BNF
  - Semântica:
    - Informal, RTFM!

3.3.4 – Control Structures

The control structures **if**, **while**, and **repeat** have the usual meaning and familiar syntax:

```
stat ::= while exp do block end
stat ::= repeat block until exp
stat ::= if exp then block {elseif exp then block} [else block] end
```

Lua also has a **for** statement, in two flavors (see §3.3.5).

The condition expression of a control structure can return any value. Both **false** and **nil** are considered false. All values different from **nil** and **false** are considered true (in particular, the number 0 and the empty string are also true).

Sintaxe vs Semântica

- *Sintaxe* diferente, Semântica igual

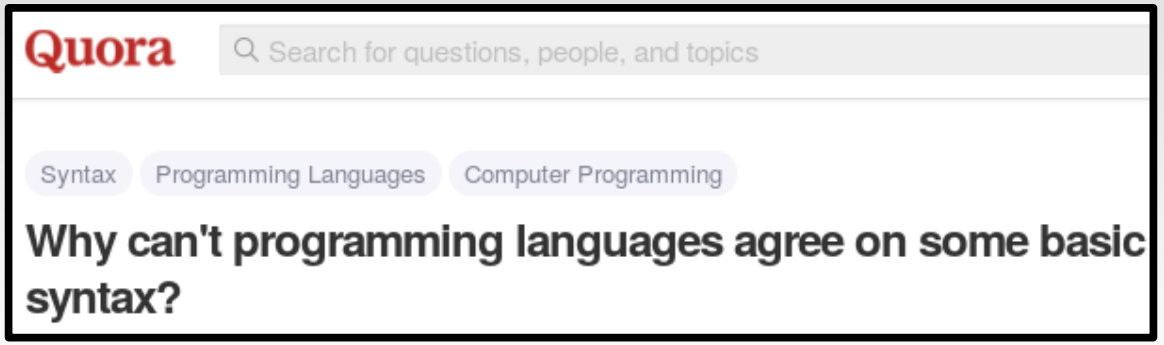
```
chico@note:~$ lua
Lua 5.1.5  Copyright (C) 1994-2012 Lua.org, PUC-Rio
> list = { 1, 2, 3 }
> print(#list)
3
>
```

```
chico@note:~$ python3
Python 3.4.3 (default, Oct 14 2015, 20:28:29)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> list = [ 1, 2, 3 ]
>>> print(len(list))
3
>>>
```

- *Sintaxe* igual, Semântica diferente

```
chico@note:~$ python2
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 1/2
0
>>>
chico@note:~$
chico@note:~$
chico@note:~$ python3
Python 3.4.3 (default, Oct 14 2015, 20:28:29)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 1/2
0.5
>>>
```

Sintaxe vs Semântica



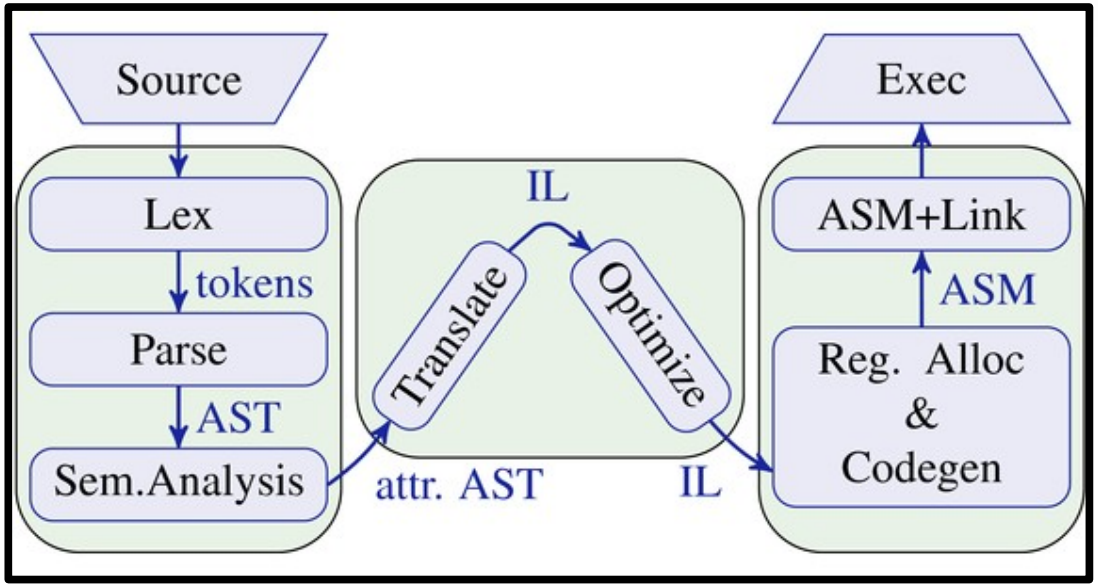
- Decisões de design:
  - indentação obrigatória (Python e Haskell)
  - opção concisa ou verbosa (Perl vs Java)
- Semântica influencia a *Sintaxe*
  - S-expressions de LISP `(+ 1 (* 5 2))`
  - Lambdas em linguagens funcionais (`\x x*2`)

Sintaxe vs Semântica

- O curso aborda, principalmente, semântica de linguagens.

Compiladores

- **Não** é um curso de compiladores.
  - Implementação vs Design



Exercícios

1. Explique com suas próprias palavras o que você entende por ***semântica*** no contexto de linguagens de programação.
2. Dê outros exemplos de abstrações que escondem detalhes da entrada, saída, memória e CPU (pelo menos um exemplo de cada).
3. Descreva a sintaxe e semântica do comando **`for`** de C.
4. Descreva a sintaxe e semântica de chamada de funções em Python.
5. Pesquise e descreva as principais diferenças semânticas entre Python 2 e Python 3.

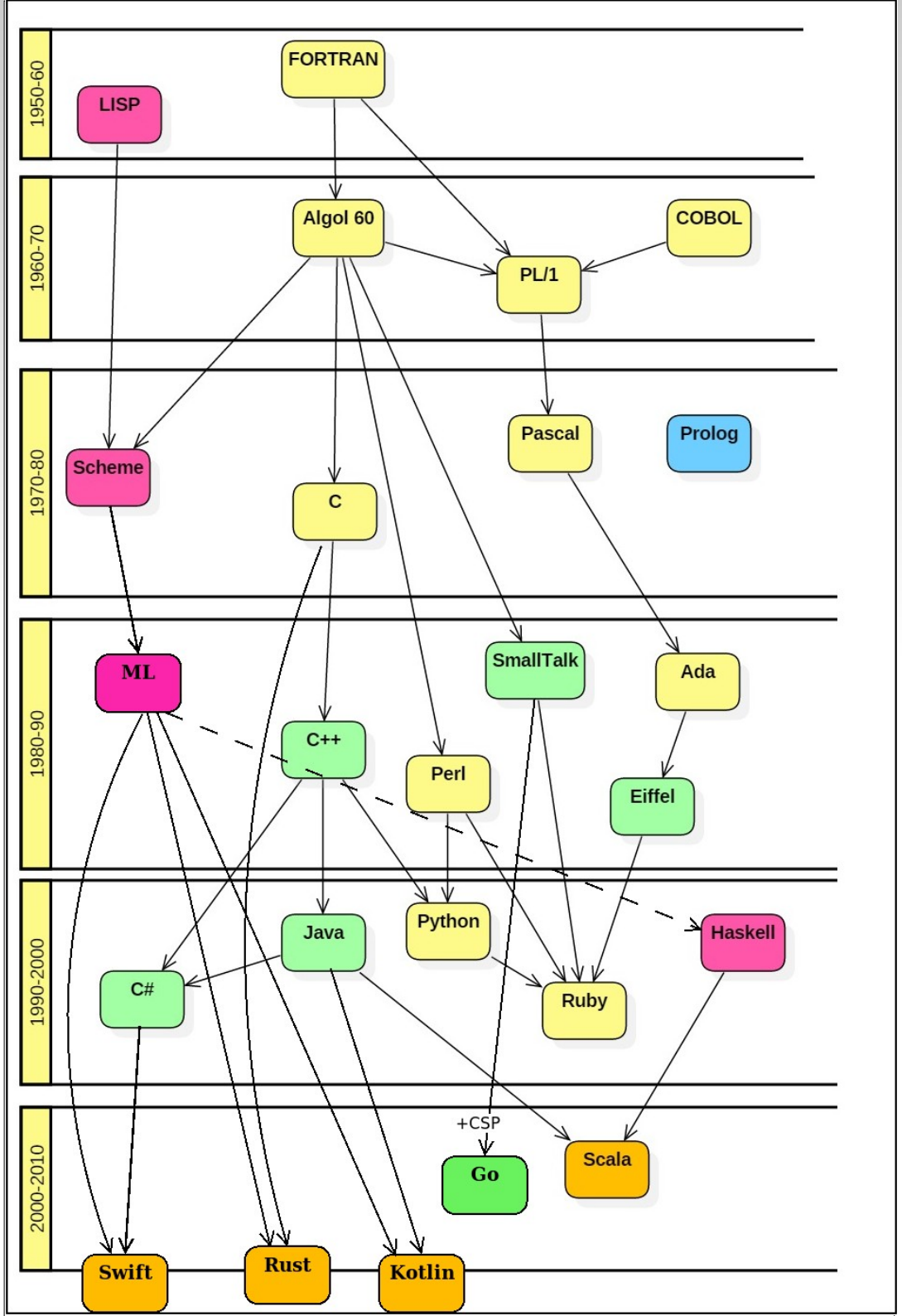
Linguagens de Alto Nível

(continuação...)

Estrutura de Linguagens

Francisco Sant'Anna

Sala 6020-B
francisco@ime.uerj.br
http://github.com/fsantanna-uerj/EDL



- Imperativa
- Funcional
- Lógica
- Orientada a Objetos

- Interpretada vs Compilada
- Dinâmica vs Estática
  - **REPL/eval**, tipagem, listas, GC

- Computação Científica
- Empresas e Negócios
- Inteligência Artificial
- Software Básico
- Redes e Protocolos
- Internet / Web
  - front/back

Exercícios

1. Pesquise e explique com suas próprias palavras a diferença fundamental entre linguagens imperativas e linguagens funcionais. Use exemplos de código.
2. Na sua opinião, por quê o paradigma de orientação a objetos se tornou tão popular?
3. Explique com suas próprias palavras a função "eval" de Python 3. Dê exemplos interessantes do seu uso.
4. Dê exemplos de outras características de linguagens dinâmicas (além das exibidas nos slides anteriores).
5. Escolha três áreas da computação (ex., computação científica, software básico, etc) e defenda o uso de uma linguagem para cada uma delas (além das já discutidas nos slides anteriores).
6. Leia o artigo “Beating the Averages” de Paul Graham e explique com suas próprias palavras o “Paradoxo de Blub”:

```
http://www.paulgraham.com/avg.html
```

Linguagens de Alto Nível

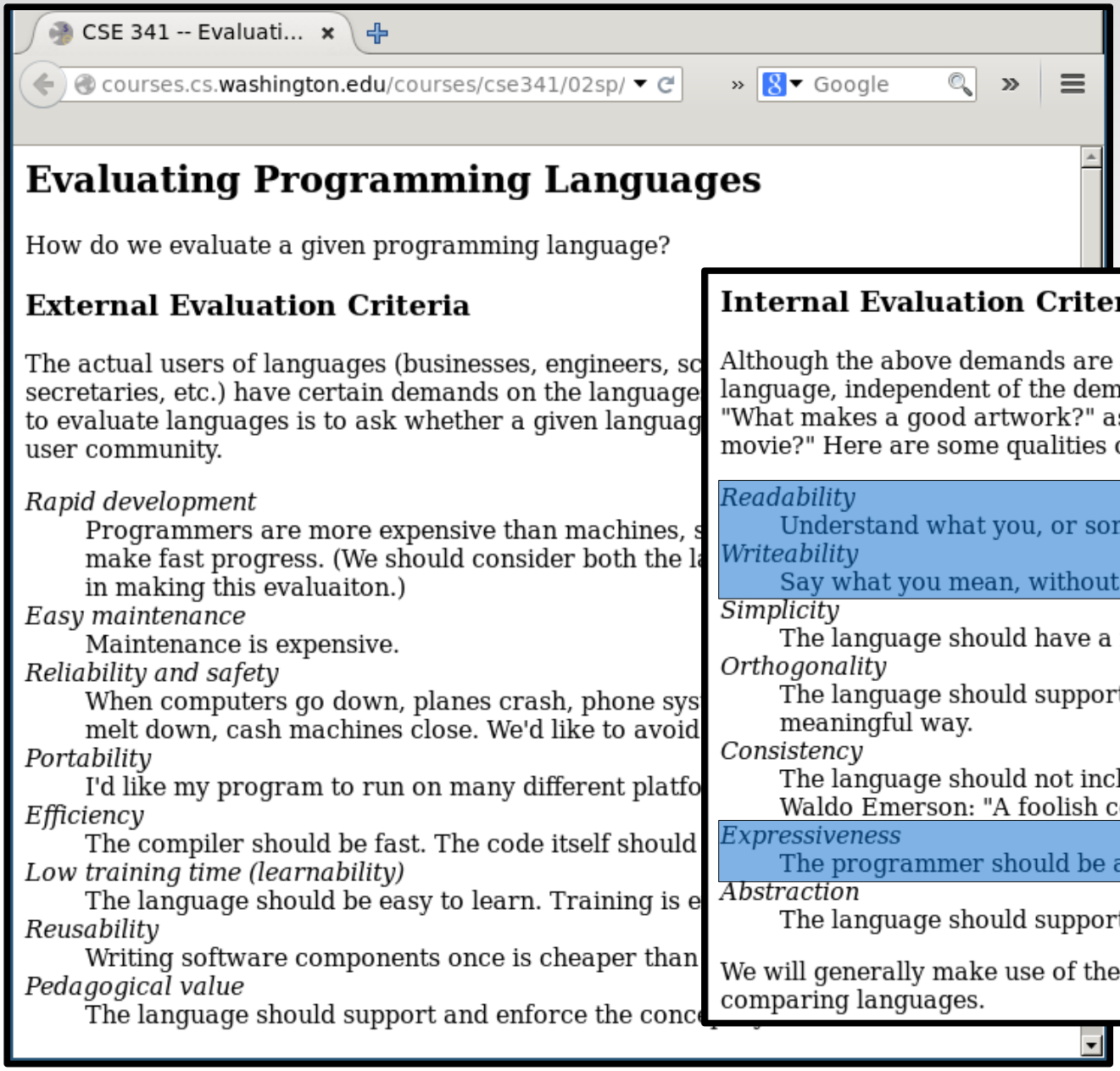
(continuação...)

Estrutura de Linguagens

Francisco Sant'Anna

Sala 6020-B
francisco@ime.uerj.br
http://github.com/fsantanna-uerj/EDL

Avaliando Linguages



Internal Evaluation Criteria

Although the above demands are all important, we should still ask what makes a *good* language, independent of the demands of its users. This is a little like the question "What makes a good artwork?" as opposed to "What makes a good Hollywood movie?" Here are some qualities of a good language.

Readability
: Understand what you, or someone else has written.

Writeability
: Say what you mean, without excessive verbiage.

Simplicity
: The language should have a minimal number of primitive concepts/features.

Orthogonality
: The language should support the combination of its concepts/features in a meaningful way.

Consistency
: The language should not include needless inconsistencies. (But remember Ralph Waldo Emerson: "A foolish consistency is the hobgoblin of small minds.")

Expressiveness
: The programmer should be able to express their algorithm naturally.

Abstraction
: The language should support a high level of data and control abstraction.

We will generally make use of these and other internal evaluation criteria when comparing languages.

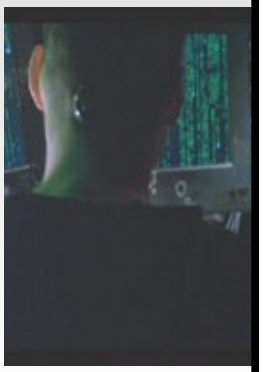
Manutenção

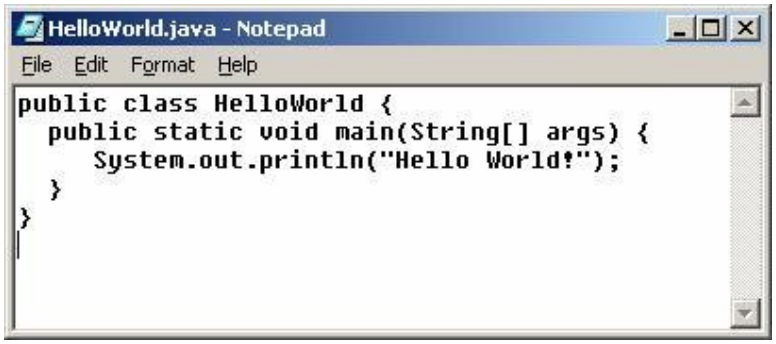
Prototipação

Readability vs Writability

```perl
while(<>) {
    split;
    print "$_[1], $_[0]\n";
}
```

```
chico@note:/data/UERJ/EDL/code$ cat names.txt
Francisco Sant'Anna
João Silva
chico@note:/data/UERJ/EDL/code$ cat names.txt | perl names.pl
Sant'Anna, Francisco
Silva, João
chico@note:/data/UERJ/EDL/code$
chico@note:/data/UERJ/EDL/code$
```





Readability vs *Writability*

```c
// C

int timeOut = 1;

<...>

timeOut = 0;
```

```java
// Java

boolean timeOut = true;

<...>

timeOut = false;
```

Exercícios

1. Descreva em linhas gerais um trabalho ou projeto que você tenha participado.
    1. Quais seriam os 2 critérios **externos** mais importantes para esse projeto? Justifique.
    2. Qual linguagem foi usada no projeto e qual seria a melhor? Justifique.
2. Descreva em linhas gerais um trabalho ou projeto que você tenha participado.
    1. Quais seriam os 2 critérios **internos** mais importantes para esse projeto? Justifique.
    2. Qual linguagem foi usada no projeto e qual seria a melhor? Justifique.
3. Escolha uma linguagem de sua preferência. Dê dois exemplos de inconsistência ou não ortogonalidade entre as funcionalidades da linguagem.
4. Pesquise dois códigos que façam a mesma coisa, mas duas linguagens diferentes, e discuta sobre a legibilidade e redigibilidade de ambos.

Linguagens de Alto Nível

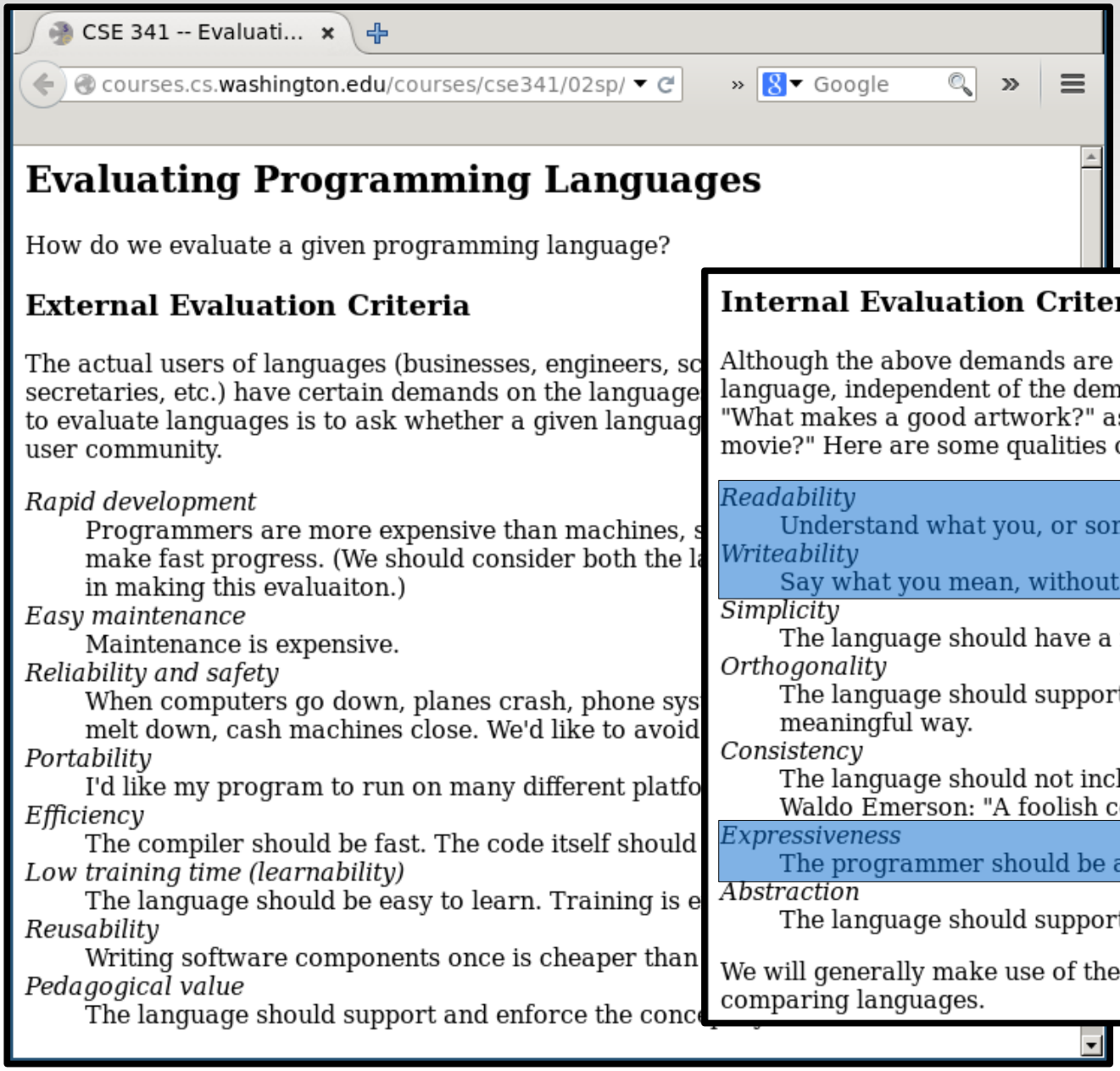
(continuação...)

Estrutura de Linguagens

Francisco Sant'Anna

Sala 6020-B
francisco@ime.uerj.br
http://github.com/fsantanna-uerj/EDL

Avaliando Linguages



Internal Evaluation Criteria

Although the above demands are all important, we should still ask what makes a *good* language, independent of the demands of its users. This is a little like the question "What makes a good artwork?" as opposed to "What makes a good Hollywood movie?" Here are some qualities of a good language.

Readability
: Understand what you, or someone else has written.

Writeability
: Say what you mean, without excessive verbiage.

Simplicity
: The language should have a minimal number of primitive concepts/features.

Orthogonality
: The language should support the combination of its concepts/features in a meaningful way.

Consistency
: The language should not include needless inconsistencies. (But remember Ralph Waldo Emerson: "A foolish consistency is the hobgoblin of small minds.")

Expressiveness
: The programmer should be able to express their algorithm naturally.

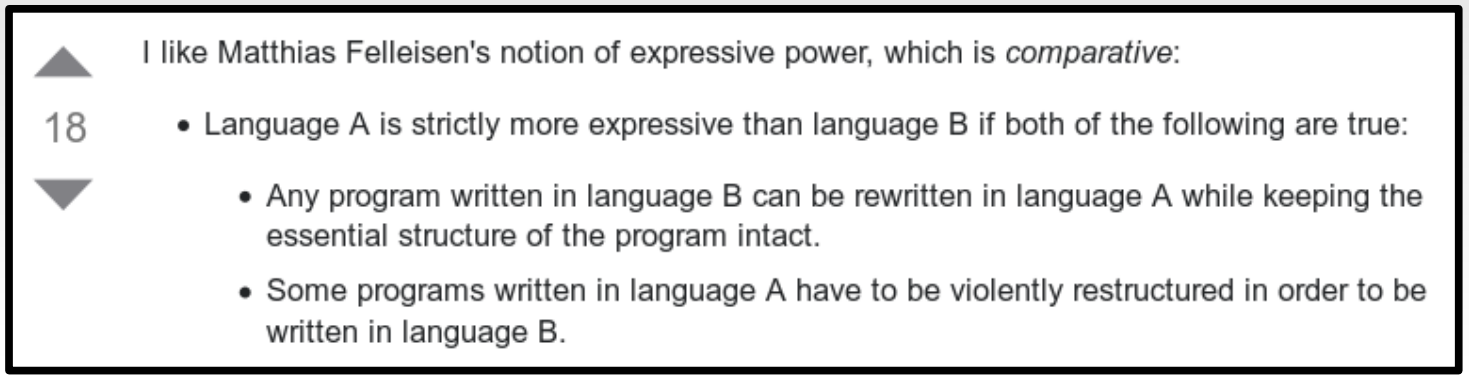
Abstraction
: The language should support a high level of data and control abstraction.

We will generally make use of these and other internal evaluation criteria when comparing languages.

Trabalho 1

- Escolher uma linguagem com a qual você **não está familiarizado**.
  - evitar duplicatas com outros colegas
  - **instalar e escrever pequenos programas com a linguagem**
  - usar pelo menos uma funcionalidade de **alta expressividade**
    - procurar uma funcionalidade "difícil" e depois sugerir/discutir com o professor
- Escrever um pequeno artigo (estilo *Wikipedia*):
  - [0.5] origens e influências (linha do tempo)
  - [0.5] classificação (imp/func/log/oo, est/din, usos)
  - [5.0] avaliação comparativa (vs outras linguagens) com foco em **expressividade**
  - [4.0] exemplos de código representativos (vs outra linguagem)
- Slides de apresentação (5-10 slides)

Poder de Expressividade



18

I like Matthias Felleisen's notion of expressive power, which is *comparative*:

- Language A is strictly more expressive than language B if both of the following are true:
  - Any program written in language B can be rewritten in language A while keeping the essential structure of the program intact.
  - Some programs written in language A have to be violently restructured in order to be written in language B.

- Usando A=*Python* vs B=*C*...
- Python é estritamente mais expressiva que C se as duas condições forem verdadeiras:
  - Qualquer programa em C pode ser reescrito em Python mantendo a **estrutura essencial do programa** intacta.
  - Alguns programas em Python precisam ser violentamente reestruturados para serem escritos em C.

Linguagens de Alto Nível

(continuação...)

Estrutura de Linguagens

Francisco Sant'Anna

Sala 6020-B
francisco@ime.uerj.br
http://github.com/fsantanna-uerj/EDL