A Language for Spoken Dialogue Management & Natural Language Processing

William Yang Wang
Xin Chen
Chia-che Tsai
Zhou Yu

Department of Computer Science, Columbia University
COMS W4115 Programming Languages and Translators

Agenda

- **Overview:** ***What is Spoke?***
- **Tutorial:** ***How to write Spoke?***
- **Implementation:** ***What is inside Spoke?***
- **Summary & Lesson Learned**
- **Demonstration:** ***Let's try Spoke!***

Where is Spoke?

Automatic Speech Recognizer

Text

Question

Spoke

Text

Answer

Speech Synthesizer

Why Spoke?

Motivation:

"It is difficult to correctly understand the intention of speakers."

"Deep linguistic analysis is needed to resolve ambiguities."

"Traditional programming language is inefficient for analyzing NL syntax."

JAVA	SPOKE
Select lib, find data, configure, train model, create structure, run, save result	All in one line. NL-specific data structure. Advanced Pattern Matching.

Language Features

Natural Language Syntax Tree

Advanced Pattern Matching

Built-in NLP APIs

Computer says,

(PRP I)
(VP (VB am)
(ADJ smart))

Simon says,

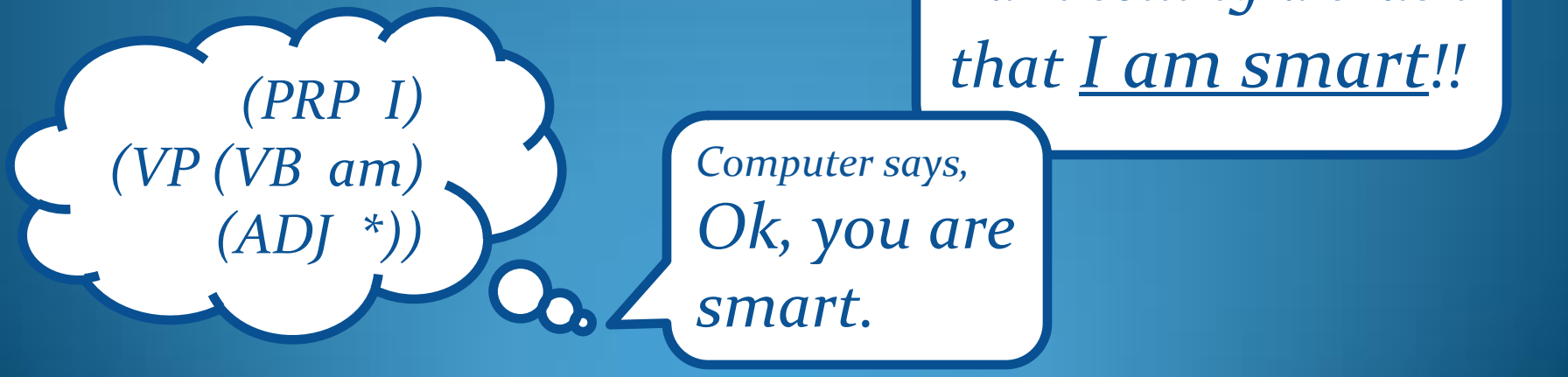
I am smart!!

Language Features

Natural Language Syntax Tree

Advanced Pattern Matching

Built-in NLP APIs



Language Features

Natural Language Syntax Tree

Advanced Pattern Matching

Built-in NLP APIs

POS Tagger,
Parser,
Chunker,
Named-entity Tagger …

Language Overview


```
 1  greeting = "Hello"
 2
 3  func print_name(name)
 4      global greeting
 5      print(greeting, name, "\n")
 6  end
 7
 8  parsed = nlpparse(readline())
 9  if parsed ~ \
10          `(N(PRP I))(V(V am)(N *))` then
11      name = myobj(match[0][1][1])
12     print_name(name)
13  fi
```

Language Overview

Language Style:

- Script Language
- Easy to write

Scoping:

- Global Variabl
- Local Variable

Domain Focus:

- Built-in API
- NLP operations

```
 1  greeting = "Hello"
 2
 3  func print_name(name)
      global greeting
      print(greeting, name, "\n")
    end
 7
 8  parsed = nlpparse(readline())
 9  if parsed ~ \
10        `(N(PRP I))(V(V am)(N *))` then
11     name = myobj(match[0][1][1])
12    print_name(name)
13  fi
```

Function

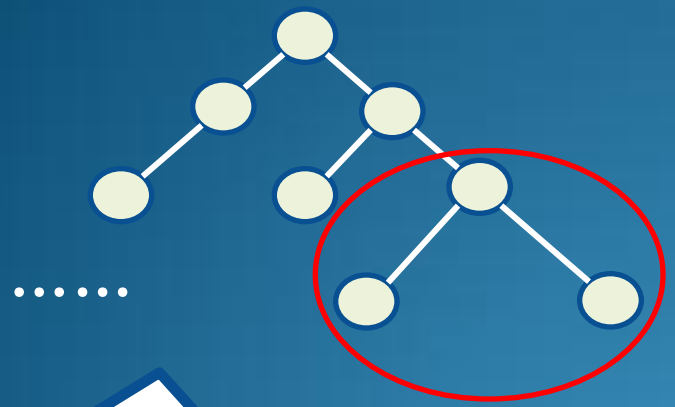
Global

Built-in API

NLP operations

Language overview

NLP operations: Syntax Tree



Standard Pattern Matching: char based
Advanced Pattern Matching: syntax based *(Spoke)*

Syntax Tree Representation

Partial Matching

```
if parsed ~ `(NP(NN *)(NN *))(VP *)` then
```

(Tag Word)

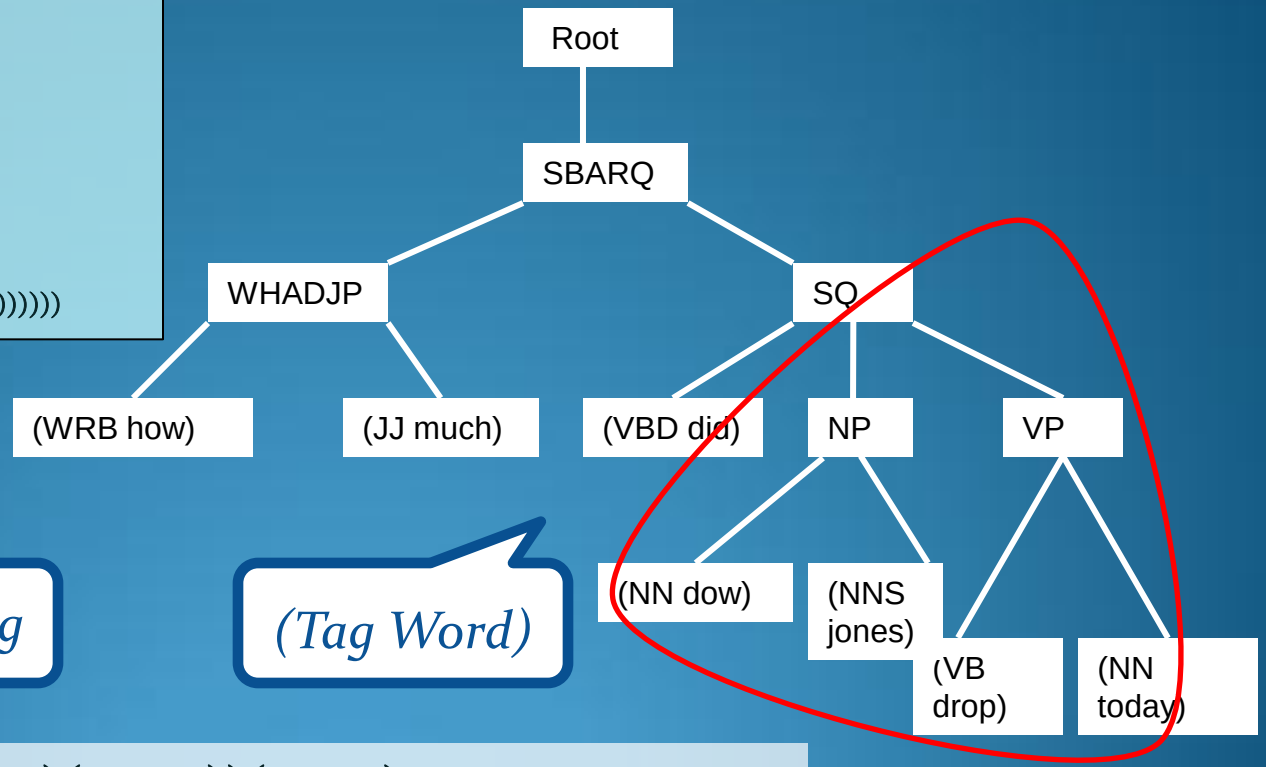
How much did Dow Jones drop yesterday?

Language overview

NLP operations: Syntax Tree

```
Input = 'How much did the Dow Jones drop
today?'
Parsed = parse (Input)

# Parsed = `(ROOT
                (SBARQ
                  (WHADJP (WRB how)
                          (JJ much))
                  (SQ (VBD did)
                      (NP (NN dow)
                          (NNS jones))
                      (VP (VB drop)
                          (NP (NN today))))))
```

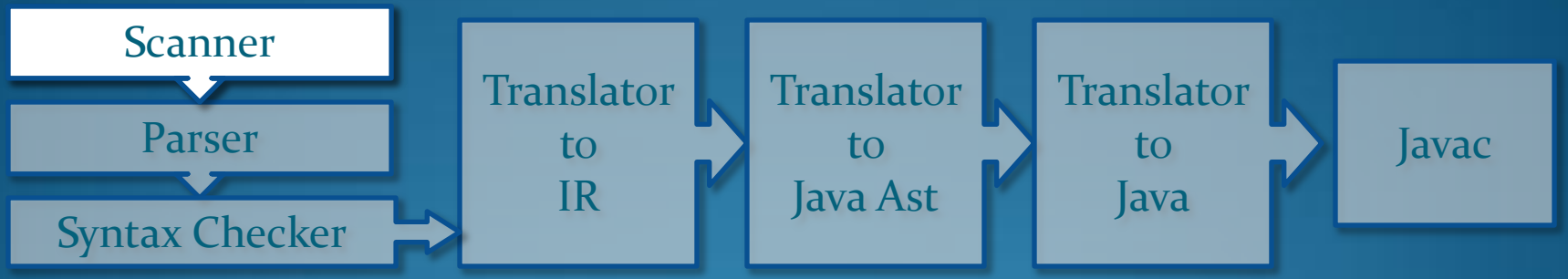


Partial Matching

(Tag Word)

```
if parsed ~ `(NP(NN *)(NN *))(VP *)` then
```

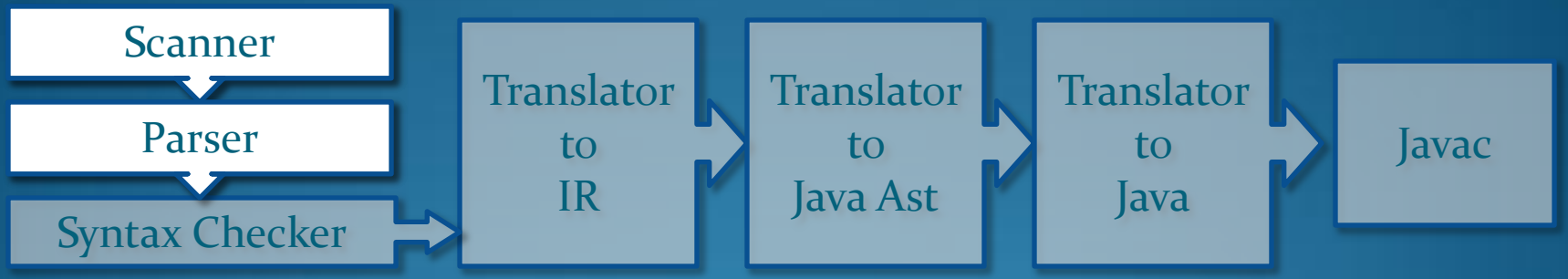
Language Implementation




```
 1  greeting = "Hello"
 2
 3  func print_name(name)
 4     global greeting
 5     print(greeting, name, "\n")
 6  end
 7
 8  parsed = nlpparse(readline())
 9  if parsed ~ \
10        `(N(PRP I))(V(V am)(N *))` then
11     name = myobj(match[0][1][1])
12     print_name(name)
13  fi
```

```
ID(greeting) ASSIGN STRING(Hello)
EOL EOL FUNCTION ID(print_name)
LPAREN ID(name) RPAREN EOL
GLOBAL ID(greeting) EOL ID(print)
LPAREN ID(greeting) COMMA ID(name)
COMMA STRING(\n) RPAREN EOL END
EOL EOL ID(prsed) ASSIGN
ID(nlpparse) LPAREN ID(readline)
LPAREN RPAREN RPAREN EOL IF
ID(parsed) BELONG LTPAREN
WORD(N) LTPAREN WORD(PRP)
WORD(I) RTPAREN RTPAREN
... ...
```

Language Implementation

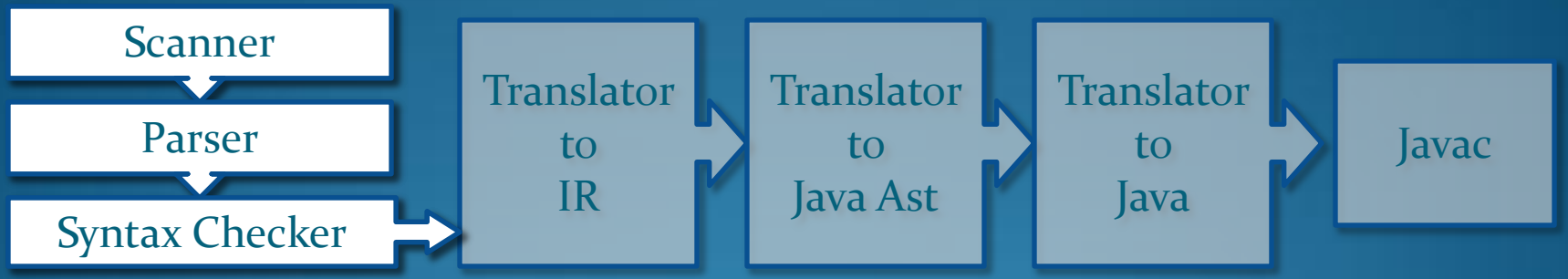



```
 1  greeting = "Hello"
 2
 3  func print_name(name)
 4      global greeting
 5      print(greeting, name, "\n")
 6  end
 7
 8  parsed = nlpparse(readline())
 9  if parsed ~ \
10          `(N(PRP I))(V(V am)(N *))` then
11      name = myobj(match[0][1][1])
12      print_name(name)
13  fi
```

```
print_name: (args: name)
  (global:greeting) (local:)
  Expr(Call(print
                Var(greeting)
                Var(name)
                String("\n")))

__global__: greeting
__local__:   parsed name
__main__:
  Expr(Assign(greeting
          String("Hello")))
... ...
```

Language Implementation




```
 1  greeting = "Hello"
 2
 3  func print_name(name)
 4      global greeting
 5      print(greeting, name, "\n")
 6  end
 7
 8  parsed = nlpparse(readline())
 9  if parsed ~ \
10          `(N(PRP I))(V(V am)(N *))` then
11      name = myobj(match[0][1][1])
12      print_name(name)
13  fi
```

Syntax Check

Continue / Break not in loops?

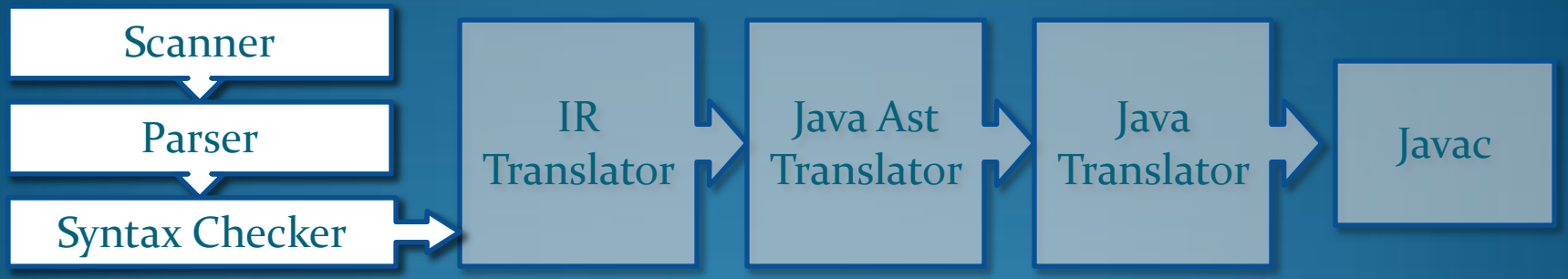
Return not in functions?

Built-in vars assigned?

... and so on

Function w/o return?

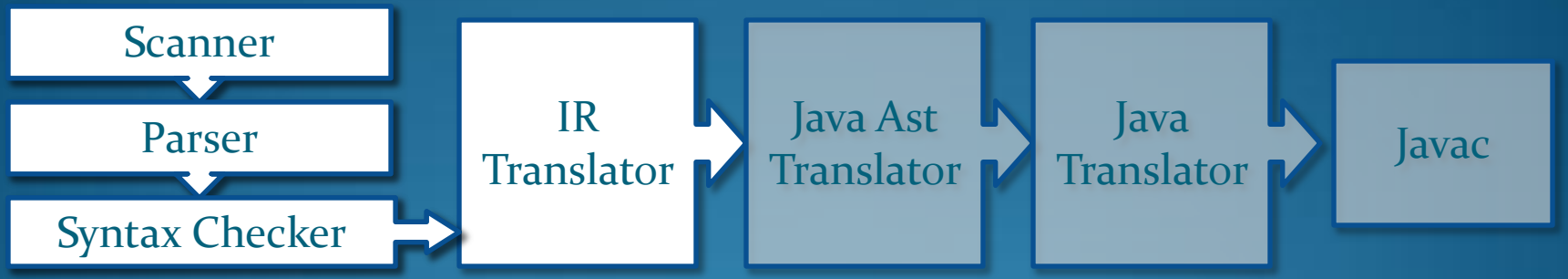
Language Implementation




```
 1  greeting = "Hello"
 2
 3  func print_name(name)
 4      global greeting
 5      print(greeting, name, "\n")
 6  end
 7
 8  parsed = nlpparse(readline())
 9  if parsed ~ \
10          `(N(PRP I))(V(V am)(N *))` then
11      name = myobj(match[0][1][1])
12      print_name(name)
13  fi
```

```
print_name: (args: name)
  (global:greeting) (local:)
  Expr(Call(print
                Var(greeting)
                Var(name)
                String("\n")))
   Return(Null)
__global__: greeting
__local__:   parsed name
__main__:
  Expr(Assign(greeting
          String("Hello")))
... ...
```

Language Implementation



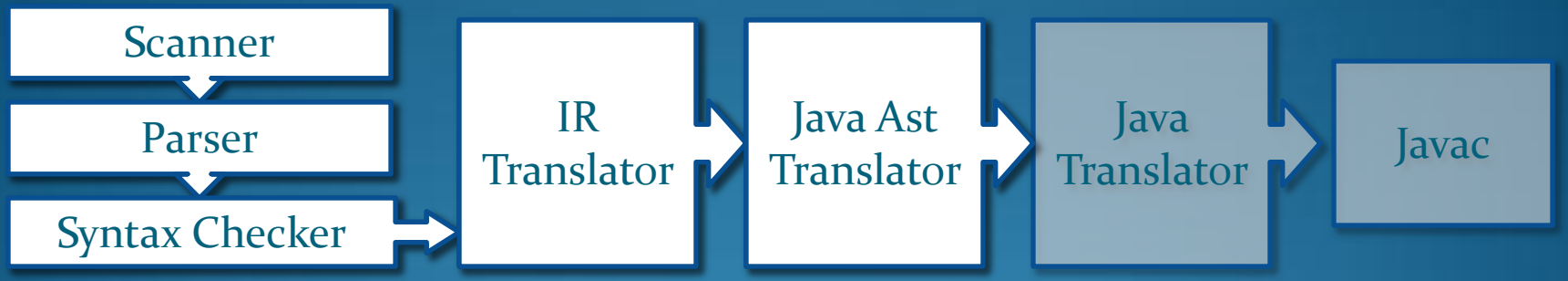

```
 1  greeting = "Hello"
 2
 3  func print_name(name)
 4     global greeting
 5     print(greeting, name, "\n")
 6  end
 7
 8  parsed = nlpparse(readline())
 9  if parsed ~ \
10        `(N(PRP I))(V(V am)(N *))` then
11     name = myobj(match[0][1][1])
12     print_name(name)
13  fi
```

```
Ll 0 1      t0 = l1
...
Hs 1 0 3    t1 = t0 ~ t3
Br 1 17 0   If(17 lines)
Wp 0 -1
Lg 1 0
...
```

```
arg(builtin)
          → l0
parsed    → l1
name      → l2
```

```
print_name
myobj  (builtin)   → f140
Match  (builtin)   → g1
star   (builtin)   → g2
greeting           → g3
```

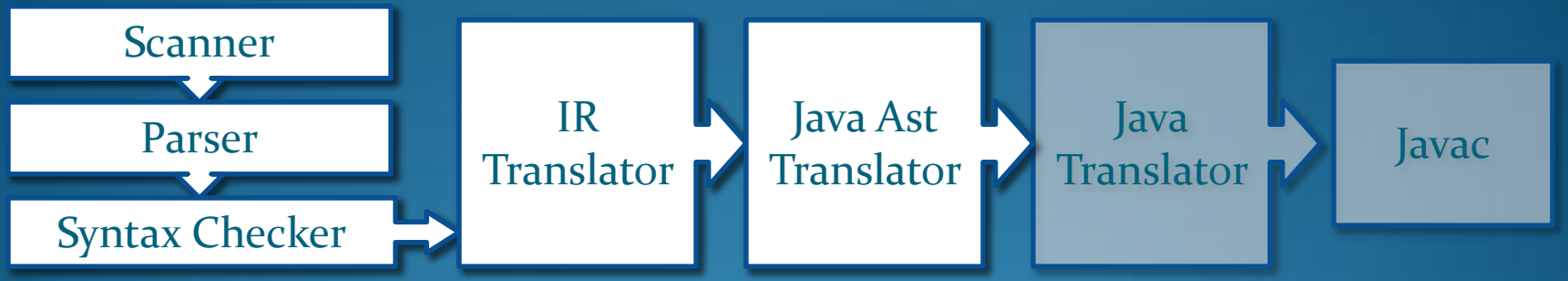
Language Implementation




```
Gl 3          Global(3)
Fn 17   f0:   Func(17)
Lc 2          Local(2)
Ll 0 0        t0=l0
...
Rt -1         Return
Lc 3          Local(3)
Cs 0 Hello    t0="Hello"
Sg 0 2        g2=t0
...
Wp 1 3        t1=[t3]
Ad 0 0 1      t0=t0+t1
Cl 1 140 0    t1=f140(t0)
Sl 1 2        t2=t1
```

```
JClass(greeting,SpokeProgram,public)
  JField(g_2,SpokeObject,private)
  JMethod(main,void,public,args:String[])
  JMethod(f_0,SpokeObject,public,l_0:SpokeObject)
    JDef(b,boolean)
    JDef(t_0,SpokeObject)
    JNew(t_0,SpokeNull) ...
    JReturn()
  JMethod(Init,void,public,l_0:SpokeObject)
    ...
    JNew(t_0,SpokeString,"Hello")
    JInvoke(g_2.Assign,t_0) ...
    t_1 = JInvoke(SpokeList.Wrap,t_3)
    t_0 = JInvoke(t_0.Operation,OpAdd,t_1)
    t_1 = JInvoke(SpokeApi.SpokeObj,t_0)   ...
```

Language Implementation

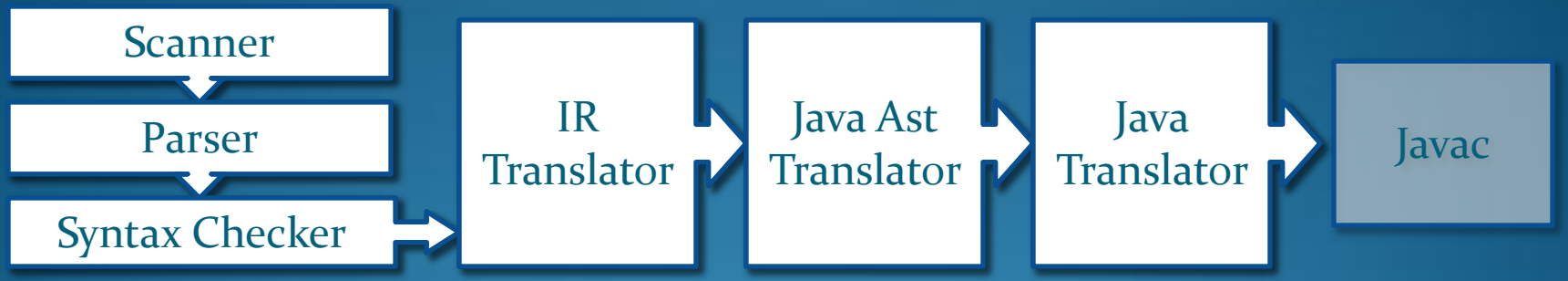



```
Gl 3          Global(3)
Fn 17   f0:   Func(17)
Lc 2          Local(2)
Ll 0 0        t0=l0
...
Rt -1         Return
Lc 3          Local(3)
Cs 0 Hello    t0="Hello"
Sg 0 2        g2=t0
...
Wp 1 3        t1=[t3]
Ad 0 0 1      t0=t0+t1
Cl 1 140 0    t1=f140(t0)
Sl 1 2        t2=t1
```

```
JClass(greeting,SpokeProgram,public)
  JField(g_2,SpokeObject,private)
  JMethod(main,void,public,args:String[])
  JMethod(f_0,SpokeObject,public,l_0:SpokeObject)
    JDef(b,boolean)
    JDef(t
    JNew(t
    JRetur
  JMethod(
    ...
    JNew(t_0,Spo          ering,"Hello")
    JInvoke(g_2.Assign,t_0) ...
    t_1 = JInvoke(SpokeList.Wrap,t_3)
    t_0 = JInvoke(t_0.Operation,OpAdd,t_1)
    t_1 = JInvoke(SpokeApi.SpokeObj,t_0)   ...
```

```
b = JInvoke(g_2.IsCompatible,t_1)
JIf(b,
  JInvoke(g_2.Assign,t_1),
  g_2 = JInvoke(t_1.Clone,))
```

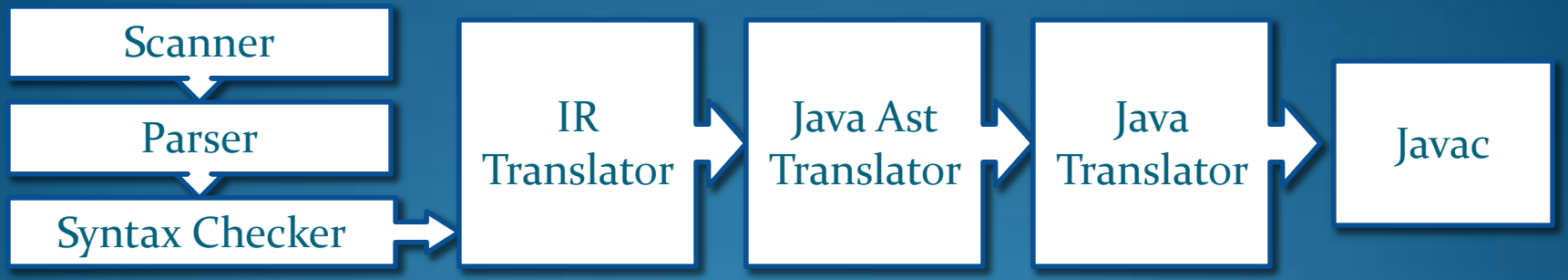
Language Implementation

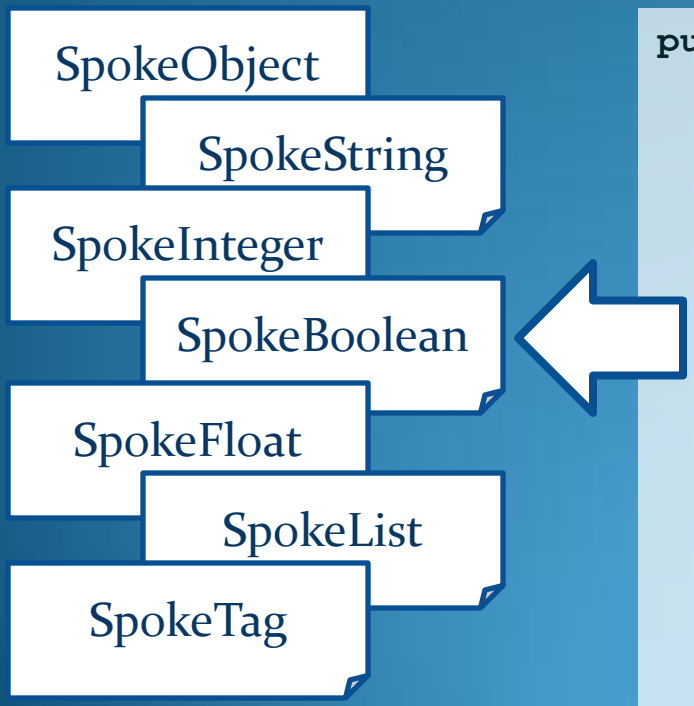



```
Gl 3          Global(3)
Fn 17   f0:   Func(17)
Lc 2          Local(2)
Ll 0 0        t0=l0
...
Rt -1         Return
Lc 3          Local(3)
Cs 0 Hello    t0="Hello"
Sg 0 2        g2=t0
...
Wp 1 3        t1=[t3]
Ad 0 0 1      t0=t0+t1
Cl 1 140 0    t1=f140(t0)
Sl 1 2        t2=t1
```

```
public class greeting extends SpokeProgram {
  private SpokeObject g_2;
  static public void main(String[] args);
  public SpokeObject f_0(SpokeObject l_0) {
    boolean b;
    SpokeObject t_0;
    t_0 = new SpokeNull();    ....
  }
  public void init(SpokeObject l_0) {
    ...
    t_0 = new SpokeString("Hello");
    b = t_1.IsCompatible(g_2);
    if(b) t_1.Assign(g_2);
    else t_1 = g_2.Clone();   ...
  }
```

Language Implementation






```
public class greeting extends SpokeProgram {
  private SpokeObject g_2;
  static public void main(String[] args);
  public SpokeObject f_0(SpokeObject l_0) {
    boolean b;
    SpokeObject t_0;
    t_0 = new SpokeNull();    ....
  }
  public void init(SpokeObject l_0) {
    ...
    t_0 = new SpokeString("Hello");
    b = t_1.IsCompatible(g_2);
    if(b) t_1.Assign(g_2);
    else t_1 = g_2.Clone();   ...
  }
```

Summary

***Spoke*, a language for Spoken Dialogues and Natural Language Processing.**

Compiler implementation from front-end (*Ocaml*) to back-end (*Java*).

Succinct representation of NLP tree

Wildcard and partial matching.

Lesson Learned

Thinking in functional language

Adaptive design for compilers

Applying PLT technique to real-life

Demonstration

The Greeting Program

A Financial Spoken Dialogue System

Thank You!