# The CEAS Programming Language

Luis Alonso (lra2103)
Hila Becker (hb2143)
Kate McCarthy (km2302)
Isa Muqattash (imm2104)

# Motivation

- Webpages contain clutter – extraneous menus, links, corporate logos, banners, etc.
- Major disadvantages for
  - visually disabled users
  - users of handheld devices with constrained screens

# Solution – Crunch

- Collection of heuristic filters that recognize and remove “clutter”
- Implementation
  - Web proxy
  - GUI
  - Document Object Model
- Avoid “one size fits all” pitfall by *adapting* to the target application

# CEAS

- A programmatic interface to the Crunch system
- Defines a simple and rich set of commands for content extraction
- Features
  - Tabbed Browsing
  - Offline Browsing
  - User defined filters via functions

# Language Characteristics

- Program flow control
  - Proceed from the first statement in a file to the last
- Data types
  - int, boolean, String, void, Page and List
- Operators
  - +, -, *, /, %, >, <, >=, <=, ==, !=, !, &, |
- Control structures
  - if, while, do while, for
- Functions
  - createPage(), extract(), append(), title(), rank(), status(), length(), print(), println(), show(), savePage()
- Extraction constants
    - IMAGES, ADS, FLASH, SCRIPTS, TXTLINKS, IMGLINKS, XTRNSTYLE, STYLES, FORMS, LINKLISTS, EMPTYTBLS, INPUT, META, BUTTON, and IFRAME

# Page Data Type

- Creating a Page

  Page createPage(“http://www.cnn.com/”);

  Page createPage(webpage);

  Page createPage(“http://www.nytimes.com/”, “pages/world/index.html”);

- Manipulating a Page

  extract(webpage, IMAGES, ADS);

  append(webpage, “next”);

  title(webpage, “My Webpage”);

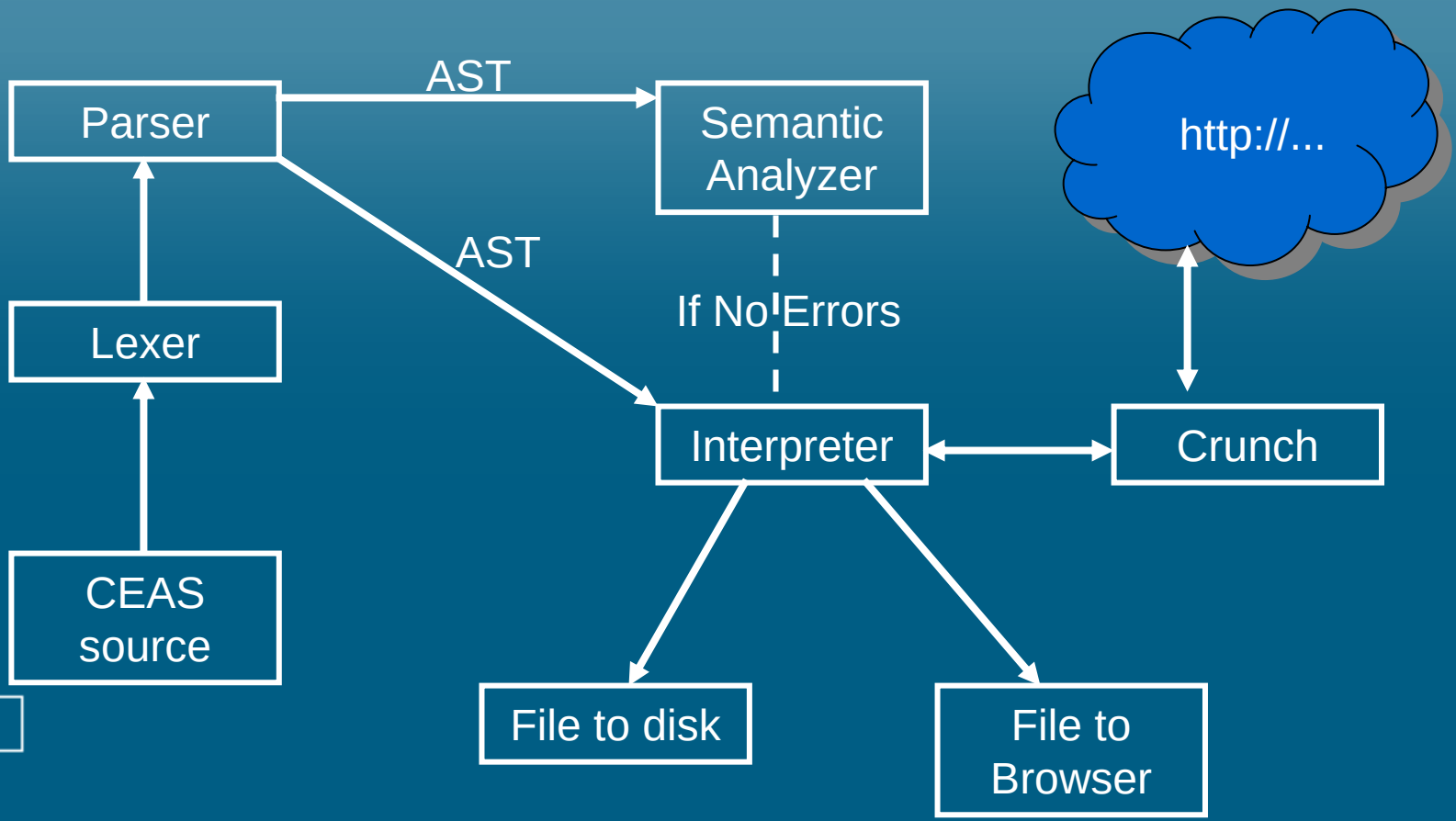
  rank(webpage, 2);

  status(webpage);

# Some Simple Examples

```
Page blog1 = createPage(“http://www.myblog.com/latestpost”);
extract(blog1, IMAGES);
show(blog1);
List pl[3];
pl[0] = createPage(“http://www.myblog.com/latestpost”);
pl[1] = createPage(”http://www.otherblog.com/latest”);
pl[2] = createPage(”http://www.newblog.com/”);
for (int i=0 : 3)
{
extract(pl[i], IMAGES);
}
show(pl);
```

# High-Level Implementation



# Implementation Details

- Semantic Analyzer
  - Uses simplified data type
  - Every line checked
- Interpreter
  - Data types actually store values
  - Performs calculations
  - Uses Crunch to extract from web pages
  - Displays HTML using built-in browser

# Implementation Details

- Functions
  - Static symbol table for functions
  - Function body, as AST, stored in table
- Variables
  - Nested symbol tables for each scope
  - Create new interpreter for included files
  - Page types are treated as Java-like objects

# Testing

- Challenges
  - LRM
  - Complexity
- What to look for
  - Code that should compile
  - Code that should not compile
- Unit Testing
  - Lexer/Parser
  - Overall Compiler

# Testing

- Methodology
  - Overall process
  - Test hardness
- Types of Tests
  - False positives
  - False negatives
  - Visual cases (title, show, rank)

# Lessons Learned

- Good planning is extremely important
- Really evaluate design before implementation
- Keep everyone in the loop
- Stick to your deadlines
- Good communication is key