The JSors Language

Diego Fulgueira, Architect
Arjun Roy, Tools
Anita Ghale, Integration
Koichiro Matsunaga, Testing
Stewin Camargo, Project Management
Stochastic process:
- A sequence of random events.
- Examples: stock prices, political elections, traffic.

Randomized Algorithm:
- An algorithm that employs randomness.
- Used to simulate stochastic processes.
Mission Statement

Goal:
- Design a language that facilitates simulation of stochastic processes.

Can be used to:
- Price financial derivatives.
- Study urban traffic.
- Calculate probabilities of all sorts.
JSors buzzwords

- Intuitive
- Event-oriented
- Flexible
- High-performance (parallelism)
- Easy to learn
- Extensible
Stochastic Process – Example #1
Gallup Daily Election Polling Results for the Democratic Presidential Nomination: Recent Trend (since Apr 4-6, 2008)

Based on national Democratic and Democratic-leaning voters

- % Clinton
- % Obama

---

Apr 06, 2008 | Apr 13, 2008 | Apr 20, 2008
Stochastic Process – Example #3
A typical JSors program...

```java
import jsors.distributions.*;

public class ChooseTest {
    public static void main(String[] args) {
        double X = N(0, 1);
        System.out.println("hello + (5 + X) * 3");
        // choose action to follow according to random value
        // of the resulting expression
        choose(X + 5) {
            [0, 10]:
                System.out.println("I'm here: [0, 10]");
                // more actions...
            [5, 10]:
                System.out.println("I'm here: [5, 10]");
                // more actions...
            break;
            default:
                System.out.println("I'm here: Default.");
                // more actions...
                break;
        }
    }
}
```

And adds JSors constructs...

Maintains usual structure of a Java program...
In addition, asynchronous random processes can also be defined...

```java
import jsors.distributions.*;

process WienerProcess
{
    double X ~ N(0, 0.2);
    state
    {
        double val = 100;
        double[] vals = new double[50];
        int i = 0;
    }
    onTick
    {
        val = val * X!;
        vals[i] = val;
        i ++;
    }
    when(i >= vals.length)
    {
        finish();
    }
}
```

```java
import org.jfree.chart.ChartFactory;

public class WienerProcessDemo extends ApplicationFrame
{
    private static final long serialVersionUID = 1631432197408679101L;

    // constructors...

    public static void main(final String[] args) throws InterruptedException
    {
        WienerProcess proc = new WienerProcess(50);
        proc.start();
        proc.join();
        final WienerProcessDemo demo = new WienerProcessDemo(array2Tin
demo.pack();
        RefineryUtilities.centerFrameOnScreen(demo);
        demo.setVisible(true);
    }
```

A JSors process declaration... can then be used in a regular Java program
General Compilation Flow:

- Lexical Analyzer (uses Java vocabulary)
- Syntactical Analyzer (uses Java grammar + JSors productions)
- Code Generator (AST walker)

JSors code

Tokens

AST

pure Java code
JSors Grammar

// JSors
processStateStatement  :  'state' '{' processStateBody '}' ;

// JSors
chooseBlockStatementGroup  :  chooseLabel blockGroup -> ^(CHOOSE_GROUP chooseLabel blockGroup) ;
JSors Benefits

- Number of lines in Java vs. number of lines in JSors
- JSors abstraction (of random process simulations)
- Easier simulation of parallel processes
- etc
Lessons Learned

- Extending a large grammar such as Java’s not an easy task.
- Should have spent more time exploring other lexer/parser generators (JFlex?)
- ANTLR requires a long learning curve.
Value Added: ANTLRWorks