Uniform General Algorithmic (UNIGA) 
Financial Trading Language

Leon Wu, Jian Pan, Jiahua Ni, Yu Song, Yang Sha

Columbia University
May 7, 2007
Outline of Presentation

- Overview of language
- Language tutorial and examples
- Architectural design and implementation
- Summary and lessons learned
UNIGA: high level scripting language for financial trading.

Language Features:
- Easy-to-use: simple syntax
- Portable: platform-independent
- Versatile: built-in functions
- Powerful: complete trading workflow
- Extendable: custom functions
Outline of Presentation

- Overview of language
- Language tutorial and examples
- Architectural design and implementation
- Summary and lessons learned
Language Tutorial and Examples

- “Hello World”
- Variables
- Loops
- “if” statement
- User defined functions
- Send an order
- Check the price
- Check the portfolio
"Hello world"

```java
main(){
    print "the market price for Microsoft is ";
    double r=market "MSFT";
    println r;
}
```

$ java Main market.uniga

the market price for Microsoft is $30.56
Variables

- Data type: “double”
- Strings are primitive
- Dates are translated via “date[]”

```plaintext
main()
{
    double d1=date[20070404];
    double d2=date[20070330];
    print "The number of days between is:";
    println d1-d2;
}
```

The number of days between is: 5.0
Loops

- "while" and "for"

```java
main(){
  double r=0;
  while(r<2){
    println r;
    r=r+1;
  }
  for(r=0;r<2;r=r+1){  println r;  }
}
```

---AST tree---
```
< main < SUBPROG < double (= r 0 ) > < while (< r 2 ) < $ SUBPROG < println r > (= r ( + r 1 ) ) > < for < FOREXPR ( = r 0 ) > < FOREXPR < < r 2 > > < FOREXPR ( = r ( + r 1 ) ) > > < SUBPROG < println r > > > > >

---End of AST---
```

```
0.0
1.0
0.0
1.0
```
main()
    double a=1, b=2;
    if a<b then{
        return 1;
    }
    else {
        return 0;
    }
}
User defined functions

- User can define their own functions
- Pass by value

```c
double increase(double r)
{
    return r+1;
}
void display(double r)
{
    println r;
    return;
}
main()
{
    display(increase(3));
}
```

```
-----AST tree-----
(FUNCDEF double increase ( DECLS ( double r ) ) ( SUBPROG
    ( return ( + r 1 ) ) ) ) ( FUNCDEF void display ( DECLS ( dou
    bles r ) ) ( SUBPROG ( println r ) return ) ) ) ( main ( SUBPR
    OG ( FUNC CALL display ( FUNC CALL increase 3 ) ) ) )

----End of AST-----

4.0
```
Send an order

- “buy” / “sell”

```c
main(){
    buy "MSFT" 1000 0 0;
    sell "INTC" 535 22.53 22.53;
}
```

Symbol ID  number of shares  stop price  limit price
An order may be filled, or discarded

<table>
<thead>
<tr>
<th>Date: 5/7/2007</th>
<th>Date: 5/7/2007</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order Type: buy</td>
<td>Order Type: sell</td>
</tr>
<tr>
<td>Stock ID: MSFT</td>
<td>Stock ID: INTC</td>
</tr>
<tr>
<td>Amount: 1000.0</td>
<td>Amount: 535.0</td>
</tr>
<tr>
<td>Stop Price: 0.0</td>
<td>Stop Price: 22.53</td>
</tr>
<tr>
<td>Limit Price: 0.0</td>
<td>Limit Price: 22.53</td>
</tr>
<tr>
<td>Filled Status: 1</td>
<td>Filled Status: 0</td>
</tr>
<tr>
<td>Filled Price: 30.56</td>
<td>Filled Price: 0.0</td>
</tr>
</tbody>
</table>
Send an order (cont’d)

- Portfolio is also changed

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Portfolio>
  <Record>
    <Date>5/3/2007</Date>
    <ID>CASH</ID>
    <Amount>-382230.0</Amount>
  </Record>
  <Record>
    <Date>5/3/2007</Date>
    <ID>MSFT</ID>
    <Amount>30500.0</Amount>
  </Record>
</Portfolio>
```
Check the price

- "high", "low", "open", "close", "volume", "market"

```c
main(){
    double op=open "MSFT" {1};
    double cl=close "MSFT" {2};
    double cu = market "MSFT";
    if op>cl then
        println cu;
}
```

-----AST tree-----
```
(main ( SUBPROG ( double ( = op ( open MSFT 1 ) ) ) ( double ( = cl ( close MSFT 2 ) ) ) ( double ( = cu ( market MSFT ) ) ) ) ( if ( > op cl ) ( println cu ) ) ) )
-----End of AST------

30.56
Check the portfolio

- “sum” – the sum of portfolio
- “pl” – the profit loss
- “holdings” – list the current positions

```c
main(){
    double pfLoss=pl();
    double assetSum = sum();
    holdings;
}
```

---AST tree---
```
< main < SUBPROG < double < = pfLoss pl > > < double < = assetSum sum > > holdings > >
------End of AST------
```

Holdings
```
Date: 5/7/2007
Stock ID: CASH
    Amount: 35826.45
Date: 5/7/2007
Stock ID: MSFT
    Amount: 6500.0
Date: 5/7/2007
Stock ID: INTC
    Amount: 1465.0
Date: 4/19/2007
Stock ID: HPQ
    Amount: 4000.0
```

---End of AST---
Outline of Presentation

- Overview of language
- Language tutorial and examples
- Architectural design and implementation
- Summary and lessons learned
Architectural Design and Implementation
### Data for Record

<table>
<thead>
<tr>
<th>Date</th>
<th>Type</th>
<th>ID</th>
<th>Amount</th>
<th>Stop</th>
<th>Limit</th>
<th>FilledStatus</th>
<th>FilledPrice</th>
</tr>
</thead>
<tbody>
<tr>
<td>5/4/2007</td>
<td>29.56</td>
<td>31.06</td>
<td>30.06</td>
<td>30.56</td>
<td>19823415</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5/6/2007</td>
<td>29.56</td>
<td>31.06</td>
<td>30.06</td>
<td>30.56</td>
<td>19823415</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Data for Record

<table>
<thead>
<tr>
<th>Date</th>
<th>Type</th>
<th>ID</th>
<th>Amount</th>
<th>Stop</th>
<th>Limit</th>
<th>FilledStatus</th>
<th>FilledPrice</th>
</tr>
</thead>
<tbody>
<tr>
<td>4/6/2007</td>
<td>buy</td>
<td>MSFT</td>
<td>500</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>28.48</td>
</tr>
<tr>
<td>5/6/2007</td>
<td>buy</td>
<td>MSFT</td>
<td>1000.0</td>
<td>0.0</td>
<td>30.5</td>
<td>1</td>
<td>30.5</td>
</tr>
</tbody>
</table>

### Data for Record

<table>
<thead>
<tr>
<th>Date</th>
<th>ID</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>5/6/2007</td>
<td>CASH</td>
<td>14310.0</td>
</tr>
<tr>
<td>5/6/2007</td>
<td>MSFT</td>
<td>4500.0</td>
</tr>
<tr>
<td>4/19/2007</td>
<td>INTC</td>
<td>2000.0</td>
</tr>
</tbody>
</table>
buy "MSFT" 1000 0 30.50;

Orders(int type, String stockID, double amount, double stopPrice, double limitPrice)

Update order (ORDERS.xml): add the order entry

stopPrice > ? limitPrice > ?

if stopPrice==0 && limitPrice==0 it is Market Order

if stopPrice>0 && limitPrice==0 it is Stop Order

if stopPrice==0 && limitPrice>0 it is Limit Order

if stopPrice>0 && limitPrice>0 it is Stop Limit Order

low<stopPrice<high

low<limitPrice<high

low<limitPrice<high low<stopPrice<high

filledStatus = 1
filledPrice = marketPrice
filledQuantity = amount

filledStatus = 1
filledPrice = stopPrice
filledQuantity = amount

filledStatus = 1
filledPrice = marketPrice
filledQuantity = amount

filledStatus = 1
filledPrice = limitPrice or stopPrice
filledQuantity = amount

Update portfolio (PORTFOLIO.xml): 1> increase/decrease cash; 2> add/update stock holding

End
<table>
<thead>
<tr>
<th>File System Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
</tr>
<tr>
<td>ParserLexer.g</td>
</tr>
<tr>
<td>Walker.g</td>
</tr>
<tr>
<td>Makefile</td>
</tr>
<tr>
<td>Main.java</td>
</tr>
<tr>
<td>—Utilities Functions—</td>
</tr>
<tr>
<td>ActivationRecord.java</td>
</tr>
<tr>
<td>CommonASTWithLines.java</td>
</tr>
<tr>
<td>Date.java</td>
</tr>
<tr>
<td>ErrorException.java</td>
</tr>
<tr>
<td>FuncScope.java</td>
</tr>
<tr>
<td>Scope.java</td>
</tr>
<tr>
<td>/data</td>
</tr>
<tr>
<td>—Built-in Functions—</td>
</tr>
<tr>
<td>Stock.java</td>
</tr>
<tr>
<td>GetRealData.java</td>
</tr>
<tr>
<td>Orders.java</td>
</tr>
<tr>
<td>Portfolio.java</td>
</tr>
<tr>
<td>—Automated Testing—</td>
</tr>
<tr>
<td>uniga.pl</td>
</tr>
<tr>
<td>unit_test.pm</td>
</tr>
<tr>
<td>bad_test_result.log</td>
</tr>
<tr>
<td>sound_test_result.log</td>
</tr>
<tr>
<td>/data/market</td>
</tr>
<tr>
<td>ACN.xml</td>
</tr>
<tr>
<td>ADBE.xml</td>
</tr>
<tr>
<td>CSCO.xml</td>
</tr>
<tr>
<td>DELL.xml</td>
</tr>
<tr>
<td>EDS.xml</td>
</tr>
<tr>
<td>HPQ.xml</td>
</tr>
<tr>
<td>IBM.xml</td>
</tr>
<tr>
<td>INTC.xml</td>
</tr>
<tr>
<td>/test</td>
</tr>
<tr>
<td>add.uniga</td>
</tr>
<tr>
<td>assign.uniga</td>
</tr>
<tr>
<td>average.uniga</td>
</tr>
<tr>
<td>builtinfunc.uniga</td>
</tr>
<tr>
<td>buy.uniga</td>
</tr>
<tr>
<td>data.uniga</td>
</tr>
<tr>
<td>division.uniga</td>
</tr>
<tr>
<td>strategy_1.uniga</td>
</tr>
<tr>
<td>while.uniga</td>
</tr>
<tr>
<td>whileandopen.uniga</td>
</tr>
<tr>
<td>whileandsell.uniga</td>
</tr>
<tr>
<td>(total 37 *.uniga files)</td>
</tr>
</tbody>
</table>
Lexer

- Defining the set of most basic tokens to be recognized by UNIGA language.
- Ex.

+, -, *, /, (), [],{}, ==, <, >, &, ","
Parser

- analyzes a sequence of tokens to determine its grammatical structure with respect to UNIGA grammar
- Left associative
- Data Type: double
- Statements: for, while, if-else, buy, sell
- Expression: open, close, high, low, volume, date
- Declaration: variable, function
Walker parses the AST and associates actions with each syntax

- Scope definition
- Operation definition
Testing

- Unit Testing, Regression Testing and Automated Testing
- Unit testing for every language construct to eliminate error at early stage
- Prepare a set of test cases, and pass all of them before uploading codes to SVN
- Deploy regression testing when a milestone is met
Outline of Presentation

- Overview of language
- Language tutorial and examples
- Architectural design and implementation
- Summary and lessons learned
Summary and lessons learned

- Team work and effective project management
  - Set up development milestones
  - Ensure on-time deliverables by regular meetings at the start of every week, constant email contacts during the week
  - Ensure team members’ understanding of weekly goal before workload breakdown.
  - Start with a small core objective and apply incremental approach in development.

- SVN (Subversion) on CUNIX
  - Source control a must for large scale team development effort
Incremental Development Approach

- Select a good application scope for the language
- Build a small core in the start, anticipate more time spent than expected at this stage
- Modularized development, separate the project into front-end and back-end
- Regression testing implemented to guarantee new features won’t break old features
Be Ready for Disasters Recovery

- You never know a single operation can cause catastrophe.
- We lost some files due to a careless operation
- Periodically back up
Thank You!

- UNIGA Team
- Columbia University
- May 7, 2007