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

Overview of Language

- 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”

```
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[]”

```
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”

```
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 ) ( S
UBPROG ( 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
```

“if” statement

```
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

```
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 ( do
uble 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”

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

Send an order (cont'd)

- An order may be filled, or discarded

```
----------------------------
Date: 5/7/2007
Order Type: buy
Stock ID: MSFT
Amount: 1000.0
Stop Price: 0.0
Limit Price: 0.0
Filled Status: 1
Filled Price: 30.56
----------------------------
```

```
----------------------------
Date: 5/7/2007
Order Type: sell
Stock ID: INTC
Amount: 535.0
Stop Price: 22.53
Limit Price: 22.53
Filled Status: 0
Filled Price: 0.0
----------------------------
```

Send an order (cont'd)

- Portfolio is also changed

```
<?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>
```

Check the price

- “high”, “low”, “open”, “close”, “volume”, “market”

```
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 ) ) ) ( doub
le ( = 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

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

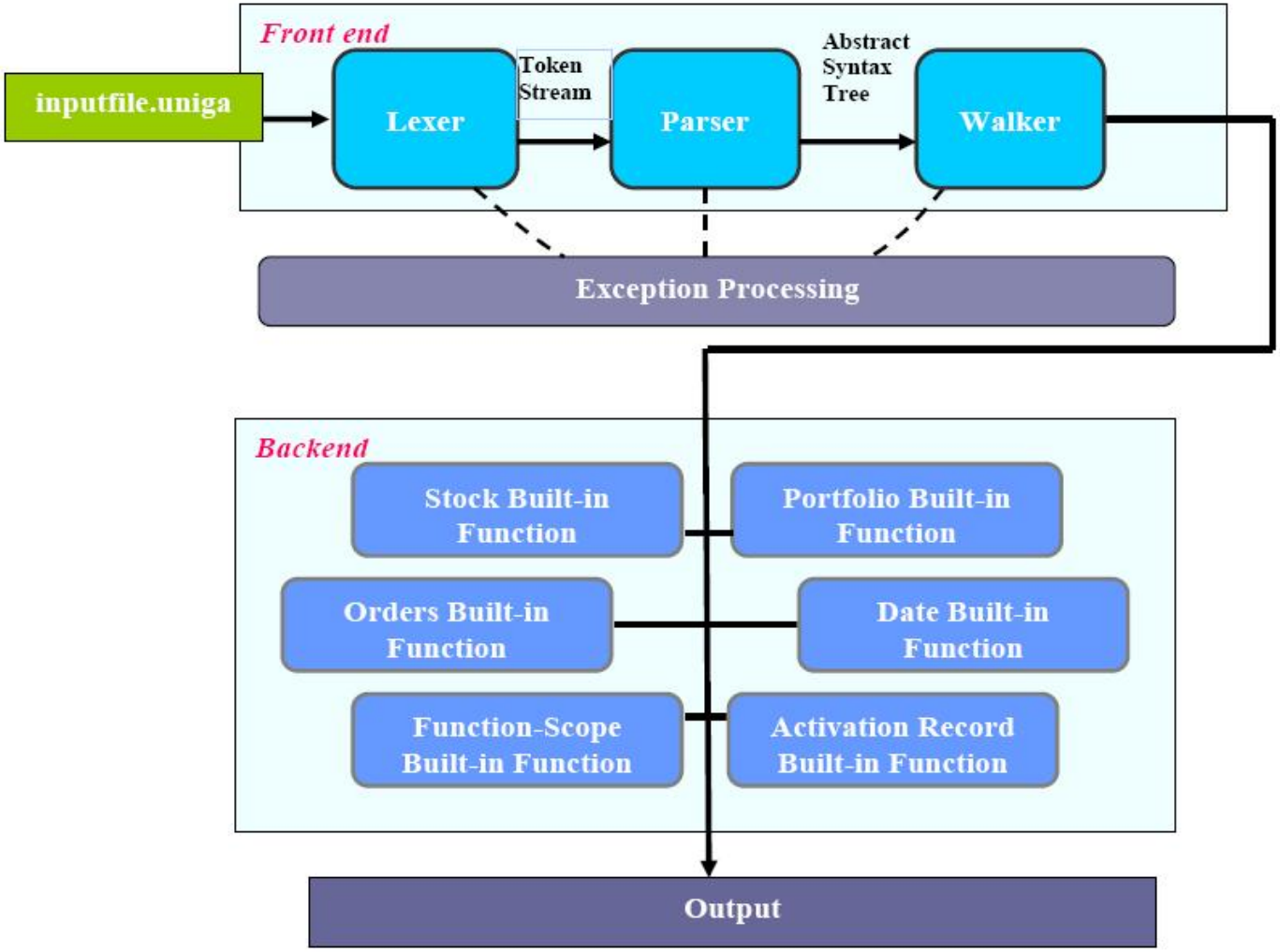
```
------AST tree------
 ( main ( SUBPROG ( double ( = pfLoss pl ) ) ( double ( = as
setSum sum ) ) holdings ) )
-----End of AST-----

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
------------------------------
```

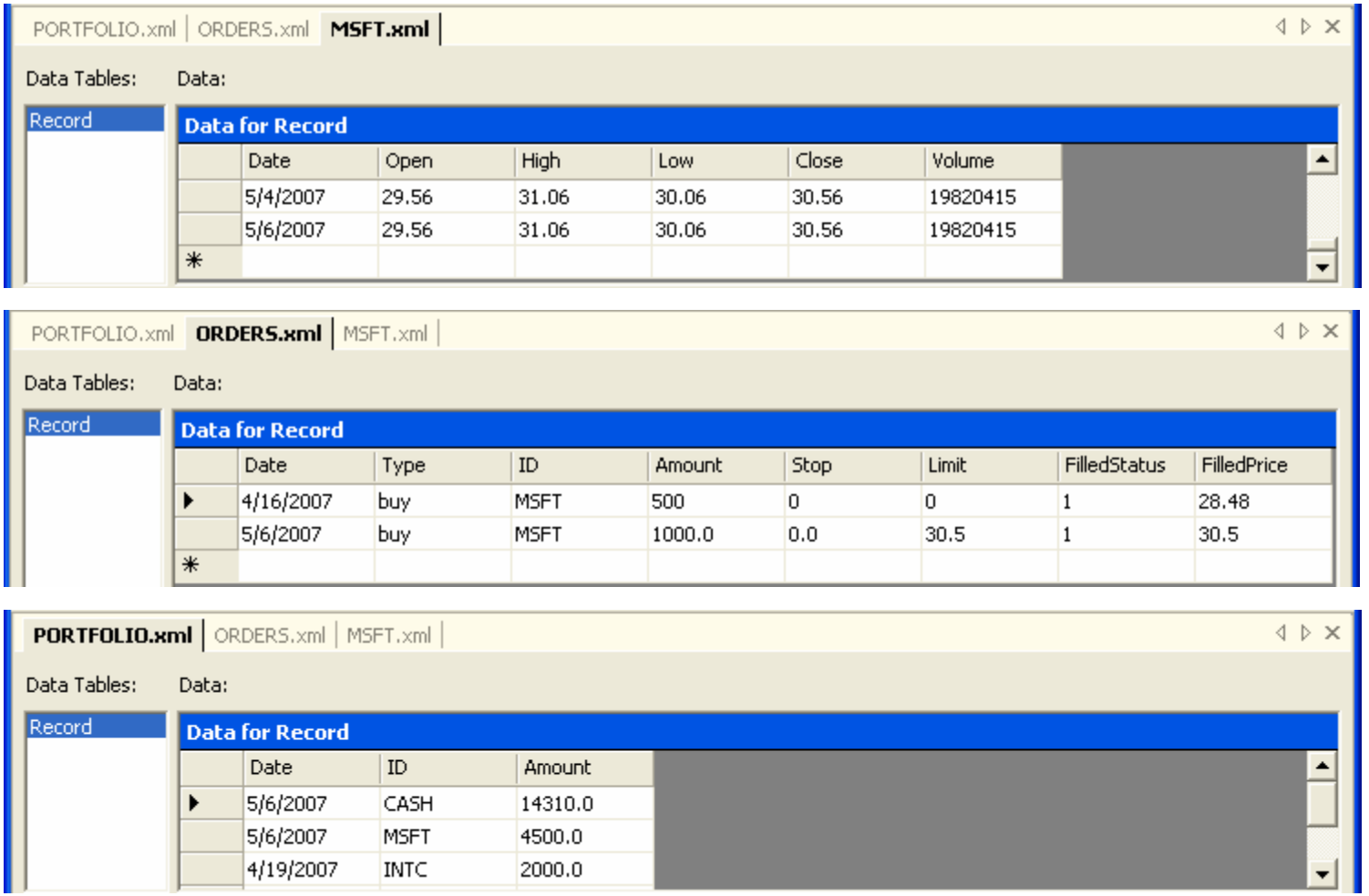
Outline of Presentation

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

Architectural Design and Implementation



Data Structure Diagram



PORTFOLIO.xml | ORDERS.xml | **MSFT.xml**

Data Tables: Record

Data:

Data for Record

	Date	Open	High	Low	Close	Volume
	5/4/2007	29.56	31.06	30.06	30.56	19820415
	5/6/2007	29.56	31.06	30.06	30.56	19820415
*						

PORTFOLIO.xml | **ORDERS.xml** | MSFT.xml

Data Tables: Record

Data:

Data for Record

	Date	Type	ID	Amount	Stop	Limit	FilledStatus	FilledPrice
▶	4/16/2007	buy	MSFT	500	0	0	1	28.48
	5/6/2007	buy	MSFT	1000.0	0.0	30.5	1	30.5
*								

PORTFOLIO.xml | ORDERS.xml | MSFT.xml

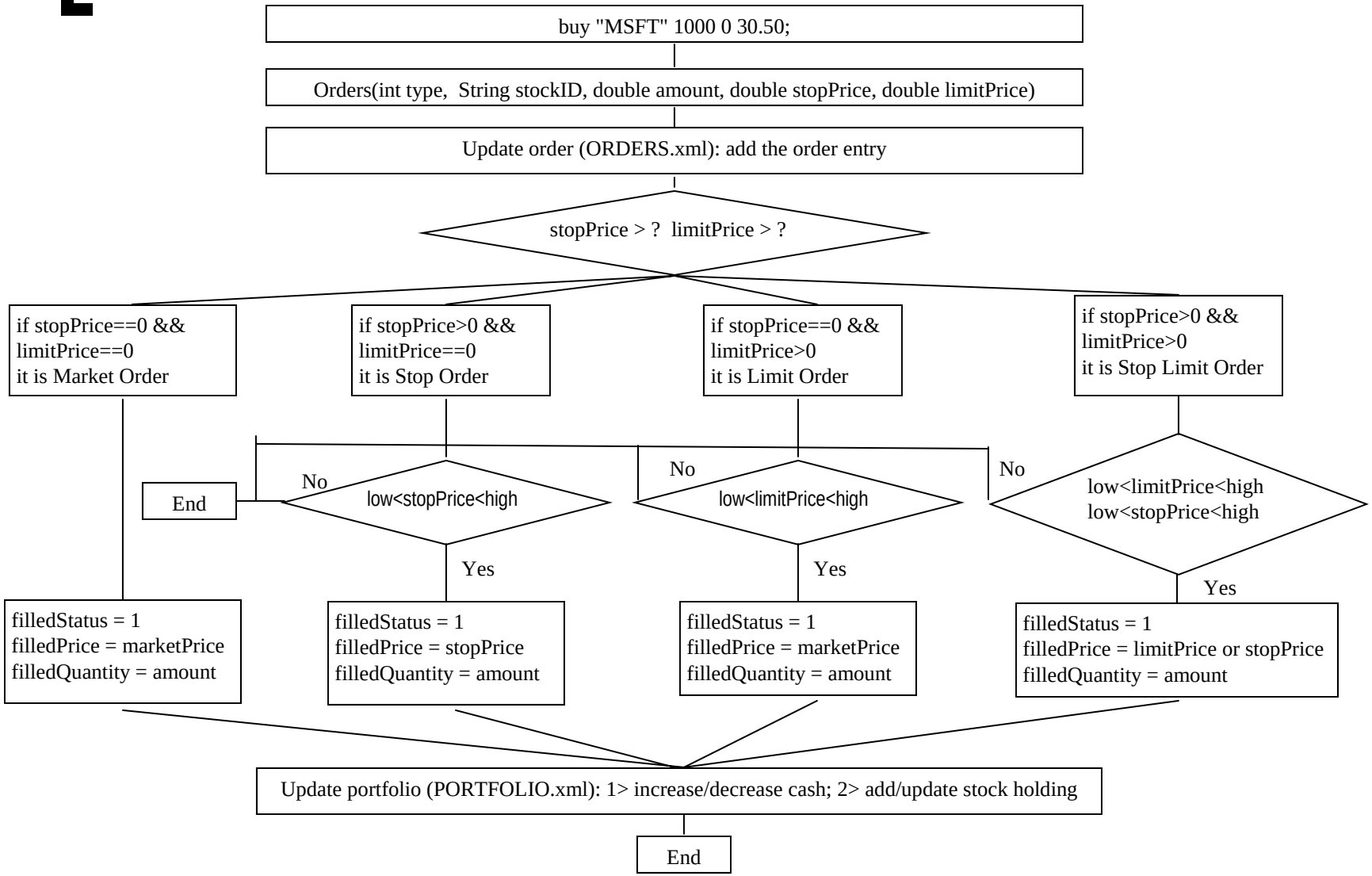
Data Tables: Record

Data:

Data for Record

	Date	ID	Amount
▶	5/6/2007	CASH	14310.0
	5/6/2007	MSFT	4500.0
	4/19/2007	INTC	2000.0

Trading Process and Data Flow



File System Structure

/
- ParserLexer.g
- Walker.g
- Makefile
- Main.java

—Utilities Functions—
- ActivationRecord.java
- CommonASTWithLines.java
- Date.java
- ErrorException.java
- FuncScope.java
- Scope.java

/
—Built-in Functions—
- Stock.java
- GetRealData.java
- Orders.java
- Portfolio.java

—Automated Testing—
- uniga.pl
- unit_test.pm
- bad_test_result.log
- sound_test_result.log

/data
- ORDERS.xml
- PORTFOLIO.xml

/data/market
- ACN.xml
- ADBE.xml
- CSCO.xml
- DELL.xml
- EDS.xml
- HPQ.xml
- IBM.xml
- INTC.xml

/test
- add.uniga
- assign.uniga
- average.uniga
- builtinfunc.uniga
- buy.uniga
- data.uniga
- division.uniga

.....
- stategy_1.uniga
- while.uniga
- whileandopen.uniga
- whileandsell.uniga
- (total 37 *.uniga files)

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

- 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