

Design and Implementation of Metalevel Architecture in C++ -- MPC++ Approach --

Outline

- Background & Motivation
- MPC++ Metalevel Architecture
- Evaluation
- Conclusion

Yutaka Ishikawa, Atsushi Hori, Mitsuhsa Sato,
Motohiko Masuda, Jorg Nolte, Hiroshi Tezuka,
Hiroki Konaka, Munenori Maeda, Kazuto Kubota

Tsukuba Research Center
Real World Computing Partnership
ishikawa@rwcp.or.jp



Background

- Traditional Programming Systems

- Language Extension Approach

- pC++, CC++, CHARM++, Mentat, FO, RTC++

How do we integrate them ?

- Library Approach

- BLAS, LAPACK, ...

Cannot optimize across library calls

```
Y = A*B + C;
```

```
MatMul(TMP, A, B);  
MatAdd(Y, C, TMP);
```

- Portable Programming Environment

- Runtime Library Specification

Degrading efficiency on some machines

**Metalevel Architecture
at Compile-time**

Users of Metalevel Architecture at Compile-time

- **Language/Library Designers**
 - **Extension/Modification/Restriction**
 - **Optimization**
 - **Source to Source Code Transformation**
- **Implementors**
 - **Open Runtime Environment Specification to port the system to any underling mechanism**

What kind of Extensions ? (1/2)

- Language Designers Perspective

- New Type/Declarator Modifier and its Operations
- New Structural Type and its Operations
- New Expressions and Statements
- New Function Capabilities

```
sync int    si;  
int * global g1;
```

```
Collection C : public Kernel {  
    ...;  
public:  
    ...;  
MethodOfElement:  
    ...;  
};
```

```
active class ac {  
    ...  
public:  
    void  
    push(int) when(buffer != FULL);  
};  
  
foo(ac *p)  
{  
    p->push(10);  
}
```

```
parallel void foo() {  
    //  
}
```

```
par {  
    // statements  
};
```

What kind of Extensions ? (1/2)

- Library Designers Perspective

**A Class Library
Designer wants to
introduce ...**

```
Mutex mm;  
Pmutex pm;  
  
foo()  
{  
    ...;  
    mm.lock();  
    ...;  
    if (cond) { mm.unlock(); return; }  
    ...;  
    mm.unlock();  
  
    pm.lock();  
    ...; // critical section  
    pm.unlock();  
}
```

```
#include "Mutex.h"  
Mutex mm;  
Pmutex pm;  
  
foo()  
{  
    ...;  
    mutex(mm) {  
        ...;  
        if (cond) return;  
        ...;  
    }  
  
    mutex(pm) {  
        ...;  
    }  
}
```

What kind of Optimization ?

```
class Matrix {  
    double data[MAX][MAX];  
public:  
    Matrix operator+(Array&);  
    Matrix operator*(Array&);  
};  
  
Matrix a, b, c, d;  
main()  
{  
    ...  
    a = b + c * d;  
}
```

```
tmp = c * d; // two loops  
tmp = b + tmp; // two loops  
a = tmp; // copy
```



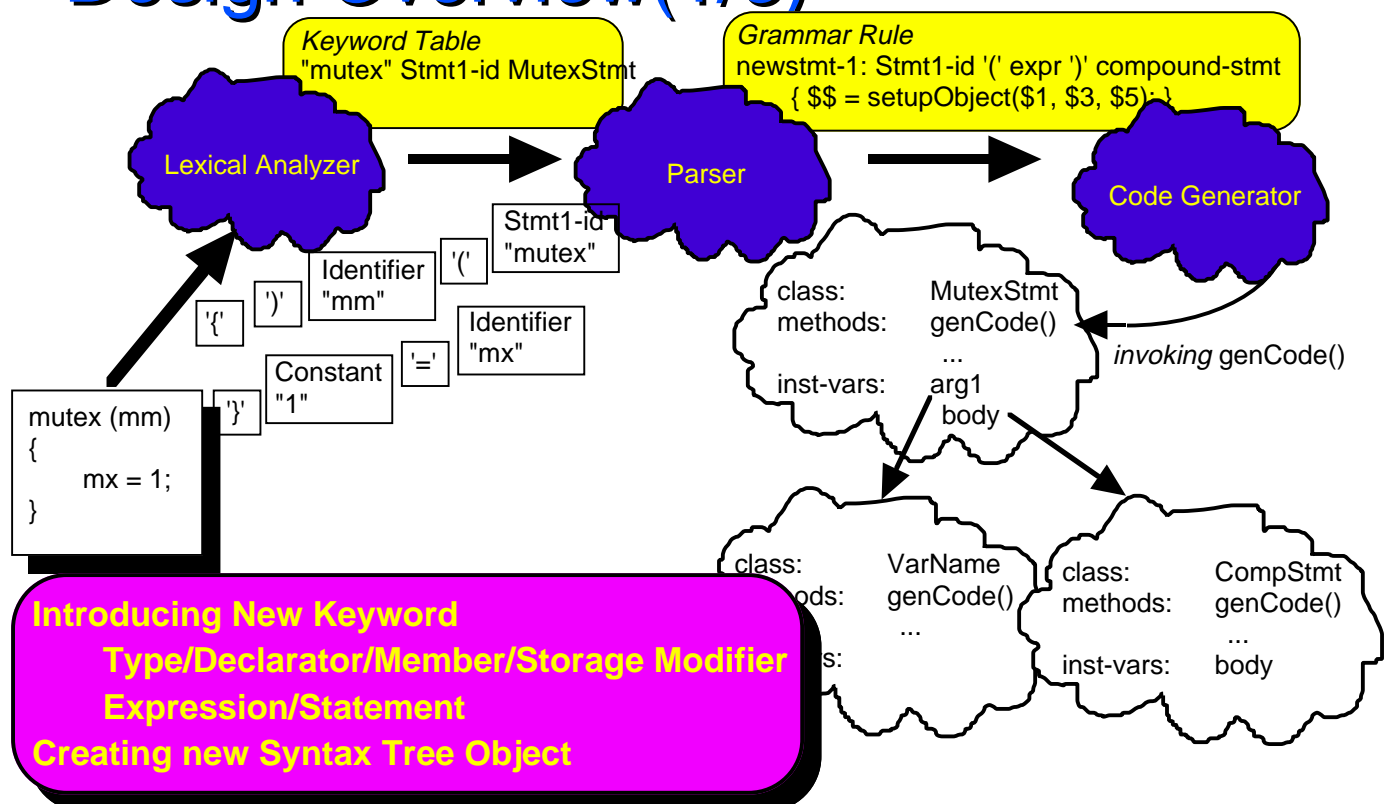
```
for (int i = 0; i < MAX; i++) {  
    for (int j = 0; j < MAX; j++) {  
        double tmp;  
        for (int k = 0; k < MAX; k++) {  
            tmp += c.data[i][k] * d.data[k][j];  
        }  
        a[i][j] = b[i][j] + tmp;  
    }  
}
```

Design Principle in MPC++

- **Applicability**
 - **Most C++ Extensions are handled**
- **Programmability**
 - **Facilities to support easy code generation description are provided**
- **Resuability**
 - e.g. the mutex statement is modified without modification of the mutex statement implementation so that number of critical sections are counted and shown at the compile-time.
- **Dynamicity**
 - **Metalevel programs are defined and used at the compile-time**

```
#include "Mutex.h"  
#include "Global.h"  
  
/* The user can use  
 * mutex statement  
 * and global pointer */
```

MPC++ Metalevel Architecture Design Overview(1/3)



Introducing New Keyword
Type/Declarator/Member/Storage Modifier
Expression/Statement
Creating new Syntax Tree Object

MPC++ Metalevel Architecture Design Overview(2/3)

- New Syntax Introduction

```
$syntaxdef $modifier $decl      global GlobalSpecModifier : public  
                                DeclModifier { ... }
```

- Modification

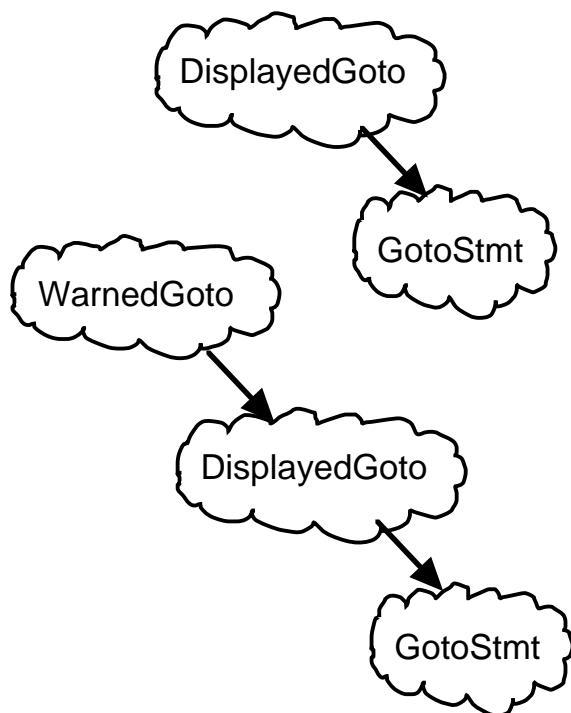
```
$meta class DisplayedGoto : public GotoStmt {  
    public:  
        int      code(OStream); };  
$meta DisplayedGoto::code(OStream os)  
{  
    ` { printf("Executing goto statement\n"); }.genCode(os);  
    return GEN_CONT;  
}  
$meta { pushKeyBind("goto", GOTO_STMT, typeid(DisplayedGoto)); }
```

- Macro Expansion

```
` { int $1=10; printf($2, $1); }.genCode(os. sym("i"), "%d");
```

Design Overview(3/3)

- Code Generation Stack



```
$meta class DisplayedGoto : public GotoStmt {
public:
    int         code(OStream); };
$meta DisplayedGoto::code(OStream os)
{
    { printf("Executing goto statement\n"); }.genCode(os);
    return GEN_CONT;
}
$meta { pushKeyBind("goto", GOTO_STMT,
                    typeid(DisplayedGoto)); }
foo()
{
    ll:
    ...
    goto ll;
}
$meta class WarnedGoto : public GotoStmt {
public:
    int         code(OStream); }
$meta WarnedGoto::code(OStream os)
{
    warningMessage("It's not better to use goto stmt\n");
    return GEN_CONT;
}
$meta { pushKeyBind("goto", GOTO_STMT,
                    typeid(WarnedGoto)); }
bar()
{
    ll:
    ...
    goto ll;
}
```

MPC++ Metalevel Architecture Design Overview(4/4)

- Context Dependent Processing

Mutex.h

```
#include "Mutex.h"
Mutex mm;
Pmutex pm;

foo()
{
    ...;
    mutex(mm) {
        ...;
        if (cond) return;
        ...;
    }
    return;
}
```

```
$meta class MutexReturn : public ReturnStmt {
public:
    int code(OStream);
};

$meta class MutexStmt : public NewStmt1 {
private:
    Label *exitLabel;
    Symbol *mutexp;
public:
    int code(OStream);
    int enter() {
        pushKeyBind("return",STMT_RETURN,
                    typeid(MutexBreak); }

    int leave() {
        popKeyBind("break", STMT_BREAK); }
};
$meta { pushKeyBind("mutex", NEWSTMT1,
                    typeid(MutexStmt)); }
```

Simple Mutex Statement Implementation

```
$meta class MutexStmt : public NewStmt1 {
    Label *exitLabel;
    Symbol *mutexp;
public:
    int code(OStream);
};
$meta { pushKeyBind("mutex", NEWSTMT1, typeid(MutexStmt));
}
$meta MutexStmt::code(OStream os)
{
    if (arg1->getType() == base_typeid(Mutex)
        || arg1->getType() == base_typeid(Pmutex)) {
        { Mutex *$1; }.genCode(os, mutexp);
        { $1 = ($2).enter(); }.genCode(os, mutexp, arg1);
        body->genCode();
        { $1->leave(); }.genCode(os, mutexp);
        { $1: }.genCode(os, exitLabel);
    } else if (arg1->getType() == base_typeid(Mutex*)) {
        ...
    } else {
        errorMessage("illegal mutex operand\n");
    }
    return GEN_END;
}
```

```
Mutex mm;
Pmutex pm;

main()
{
    ...;
    mutex(mm) {
        ...; // mutual exclusion body
    }
    mutex(pm) {
        ...; // mutual exclusion body
    }
}
```



```
Mutex *_SYM01;
_SYM01 = (mm).enter();
...; // mutual exclusion body
_SYM01->leave();
Mutex *_SYM02;
_SYM02 = (pm).enter();
...; // mutual exclusion body
_SYM02->leave();
}
```

MPC++ Portability

- MPC++ Portable Runtime Library
- Machine Dependent Runtime Library

```
#include "i486.h"
extern int foo(int, int);
main()
{
    int i;
    entry(int) l1;

    i = foo(1, 2)@[5];
}
```

```
#include "sunos.h"
extern int foo(int, int);
main()
{
    int i;
    entry(int) l1;

    i = foo(1, 2)@[5];
}
```

i486.h

```
$synobjdef generalFuncExpr : FuncExpr {
public:
    int code(OStream);
};
$meta {
    objbind(EXPR_FUNC);
};
$meta generalFuncExpr::code
{
    ...
    if (isMpcFuncExpr) {
        cntname = continuation->getName();
        tkn = gensym("_tkn_", cntname);
        retval = gensym("_retval_", cntname);
        entarg = gensym("_entargs_", cntname);
        func = gensym("_threaded_", function->getName());
        `( ($1 = _mpclnitEntry(&($2), (char*)&$3,
                             sizeof(struct $4))),
          _mpcRemoteAsyncInvoke($5, $6, (char*)0, 0, $1) ).
          genCode(os, tkn, continuation, retval, entarg, remote,
                 func);
        return GEN_END;
    }
    ...
}
```

This C++ Code may be replaced with a code for the target machine

MOL (Metalevel Object Library)

- **Extensions**
 - **Collection**
 - **Global Pointer**
 - **Persistent Object**
 - **Mutex Statement**
- **C++ Class/Template Interface is provided**
 - **Its Implementation is defined in Metalevel**
 - **The library user can optimize his library in metalevel**

MOL (1/2)

- Extensions

- Collection
- Global Pointer
- Persistent Object
- Mutex Statement

```
#include "pcxx.h"
Collection Linearset; public
    Kernel { ... };
class MyElement { ... };

foo()
{
    LinearSet<MyElement> cc;
    cc.hello();
}
```

```
#include "Global.h"
float
foo(float *global gp)
{ return *gp/mype; }
mpc_main()
{
    float    f = 2.0;
    foo(&f)@[5];
}
```

```
#include "PersistentObject.h"
persistent class PC {
    int pc1;
    char *pc2;
};
main()
{
    Storage *st;
    st = new Storage("test.po", "r");
    p1 = storage->findObject("p1");
}
```

```
#include "Mutex.h"
Mutex    mm;
main()
{
    ...;
    mutex(mm) {
        ...; // critical section
    }
}
```

MOL(2/2)

```
class Matrix {
  double data[MAX][MAX];
public:
  Matrix operator+(Array&);
  Matrix operator*(Array&);
};

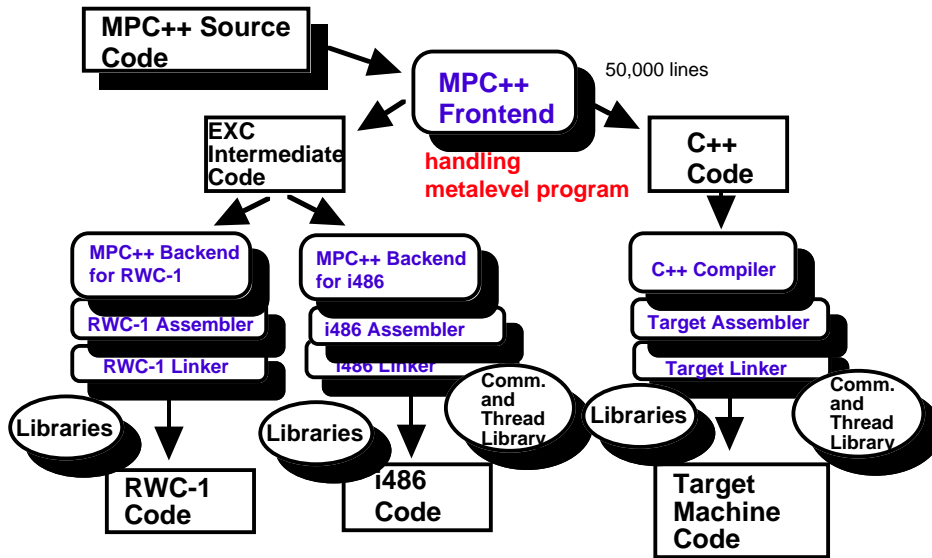
Matrix a, b, c, d;
main()
{
  ...
  a = b + c * d;
}
```

```
for (int i = 0; i < MAX; i++) {
  for (int j = 0; j < MAX; j++) {
    double tmp;
    for (int k = 0; k < MAX; k++) {
      tmp += c.data[i][k] * d.data[k][j];
    }
    a[i][j] = b[i][j] + tmp;
  }
}
```

Source Code Transformation
Description in the metalevel



MPC++ Language System



MPC++ Application				
MPC++ULT				RWC-1
PM	UDP/IP	AP1000	CM-5	
Myrinet	Ethernet	<small>(Yonezawa Lab.)</small>		

Evaluation (1/2)

- **Basic Performance**

	Meta(usec)	C++(usec)	Relative
for statement	167	0.179	933
call/return	128	0.095	1,347
a = b + c	193	0.061	3,163
object creation	195	3.530	55

Evaluation (2/2)

- Metalevel Processing

```
$meta clas MyAsgnExpr:AsgnExpr {  
  public:  
    int    code(OStream);  
};  
$meta MyAsgnExpr::code(OStream)  
{ return GEN_CONT; }  
$meta { objBind("=", ASGN_EXPR,  
              MyAsgnExpr); }
```

```
$meta clas MyAsgnExpr:AsgnExpr {  
  public:  
    int    code(OStream);  
};  
$meta MyAsgnExpr::code(OStream os)  
{  
  ` (printf("assign expr\n"),  
     ($1 = $2 ), $1)).genCode(os,  
                               left, right);  
  return GEN_END;  
}  
$meta { objBind("=", ASGN_EXPR,  
              MyAsgnExpr); }
```

```
foo(int a, int b)  
{  
  a = b;  
}
```

	Time(usec)	Relative
Normal	233	1
Meta(Null)	627	2.7
Meta(Print)	1657	7.1

Related Work

- **OpenC++ ver2.0**
 - **Design**
 - **Small Extension in C++**
 - **Metalevel facilities are designed in the C++ class system**
 - **What kind of restrictions behind ?**
 - **C++ is not only an object-oriented language but also C !!**
 - **C statements**
 - **if, for, continue, break, goto**
 - **Metalevel Programs are compiled**

?

```
#include "Mutex.h"  
#include "Global.h"  
  
/* The user can use  
* mutex statement  
* and global pointer */
```

Concluding Remarks

- **MPC++**
 - **Metalevel Architecture at Compile-time**
 - **C++ Language Extension/Modification/Restriction**
 - **New Style of Designing Libraries**
 - **Open Implementation for parallel/distributed systems**
- **Future Work**
 - **Efficient Metalevel Program Execution**
 - **on-the-fly compilation**
 - **native compilation**
 - **Design of MOL for parallel/distributed systems**
 - **Design of Metalevel Programming Interface**
- **URL**
 - **<http://www.rwcp.or.jp/people/mpslab/mpc++>**