4th Workshop on Productivity and Performance (PROPER 2011)
Tools for HPC Application Development

at EuroPar 2011 Conference, Bordeaux/France
August 30th 2011

Automatic Source Code Transformation for GPUs based on Program Comprehension

Pasquale Cantiello and Beniamino Di Martino
Second University of Naples
Beniamino.dimartino@unina.it

Agenda

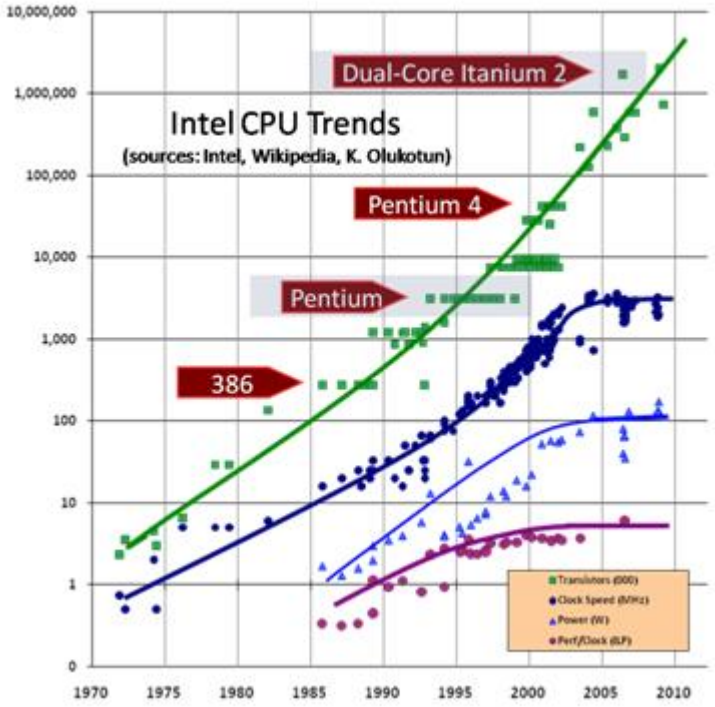
SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

- Motivations
- Algorithm recognition
- Source code transformation
- Case studies
- Future research directions

Motivations

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

- Processors' evolution over years has been mainly driven by the growing of the number of transistors per chip and by the increasing of the clock speeds.
- In the last five years, physical limits on junction dimensions and the power that must be dissipated on a single chip have posed a serious barrier to the increasing of CPU frequencies.
- Chip producers now put more than one processing unit on a single chip. Units with 8 or 16 cores are commonly sold.
- Special purpose devices as GPUs (Graphic Processing Units), with hundreths of cores can be used to do massive computation.



Motivations

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

- Computer systems, especially those designed for scientific applications have complex architectures with several interconnected computation nodes each with one or more CPUs and/or GPUs. Each device has it's own memory hierarchy. Devices are interconnected through networks that differ in topology, bandwidth and latency times.
- Different programming models, languages and parallelization approaches.
- Software needs to be ported (in some case rewritten) with high costs due to time required, very skilled developers needed and testing to new faults to software.
- Automatic transformation is needed.

NVIDIA CUDA

OpenCL

OpenMP

MPI

Algorithm Recognition

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

- In order to develop a metodology to automate the porting of software to new architectures, our starting point is the analysis of source code, not only on its structure, but also on its semantic, trying to automatically understand what a program does.

```
// Main loop.
for (int j = 0; j < n; j++) {
   double[] tempLrowj = matriceU[j];
   double d = 0.0;
   for (int k = 0; k < j; k++) {
      double[] tempLrowk = matriceU[k];
      double s = 0.0;
      for (int i = 0; i < k; i++) {
         s = s + tempLrowk[i]*tempLrowj[i];
      }
      s = (matriceA[j][k] - s)/matriceU[k][k];
      tempLrowj[k] = s;
      d = d + s*s;
   }
   d = matriceA[j][j] - d;
   matriceU[j][j] = Math.sqrt(Math.max(d,0.0));
   for (int k = j+1; k < n; k++) {
      matriceU[j][k] = 0.0;
   }
}
```

Semantic program analysis and algorithm recognition

- Source code analysis performed until structural level (Program Dependence graph, Class diagrams, ect)
- Semantic analysis (concept assignment)
- Problem impossible to solve automatically at the Application domain level
- Automatable if focused towards programming oriented concepts
- Algorithmic Recognition, through hierarchical abstraction

ALCOR: ALgorithmic COncepts Recognizer

Technique (and prototype tool) for detection of algorithmic patterns (concepts) in source code

- Hierarchical Abstraction of algorithmic concepts: Hierarchical Concept Parsing (Concept Grammar)
- Only partial recognition needed: demand driven approach
- Top-down recursive descent parsing
- Focus on structural properties (control-data dependence) for concept characterization
- Inherently delocalized structural program repres.
- Global scope of visibility for concepts

Concept Grammar

- Formalism based on Attributed grammars
- Concept Grammar CG = (G,A,R,C)
- G = (T,N,P,Z) associated context-free grammar
- Terminal symbols (T) ⇔Base Concepts
- Nonterminal symbols (N) ⇔ Concepts
- Start Symbols (Z) ⇔ Algorithmic Patterns to be recognized
- Production rules (P) ⇔Recognition rules
- V z ∈ Z language L(z,CG) generated by CG ⇔set of implementations of z
- Sentence of CG ⇔ set of statements associated to a concept
- Derivation graph ⇔ Hierarchy of composition

Recognition rules

rule `concept` →

composition

SubConcepts

condition

local attribute : type

Conditions

attribution

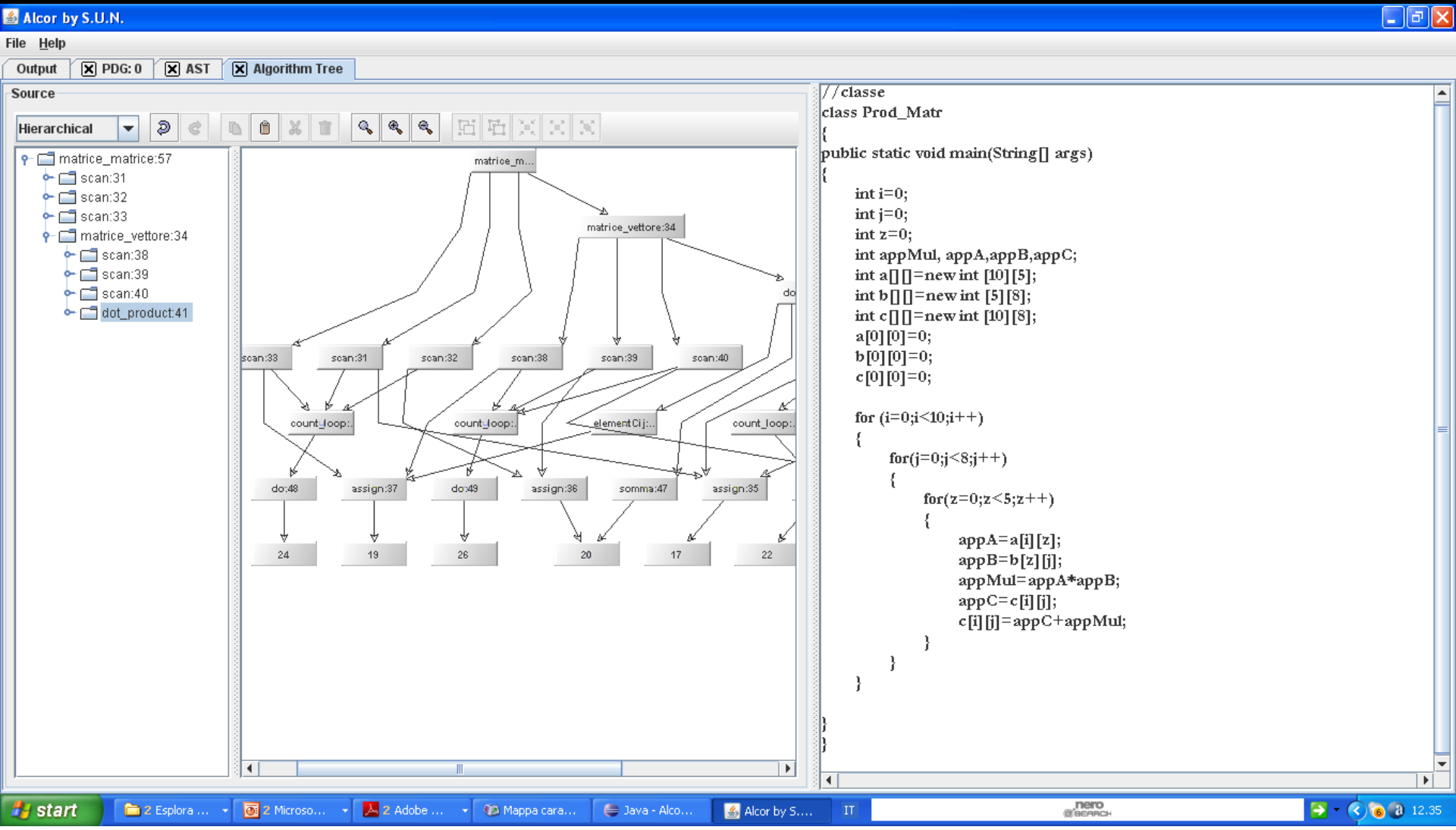
local attribute : type

AttributionRules

Example: matrix-matrix product

rule matrix_matrix_product →
composition
scan[1]
matrix_vector_product
scan[2]
condition
local countLoopList1, countLoopList2 : [hierarchy]
control_dep(matrix_vector_product,scan[1],true)
scan[1].array_scan.array_inst = matrix_vector_product.res_vector_struct.inst
scan[1].array_scan.subscr_pos ≠
matrix_vector_product.res_vector_struct.matr_vec_subscr_pos
scan[1].hier = -(-,countLoopList1,)
scan[2].hier = -(-,countLoopList2,-)
countLoopList1 = countLoopList2
scan[1].array_scan.scan_index = scan[2].array_scan.scan_index
scan[1].range = scan[2].range = whole
scan[1].stride = scan[2].stride = sweep
scan[2].array_scan.array_inst = matrix_vector_product.vector_struct.inst
scan[2].array_scan.subscr_pos ≠
matrix_vector_product.vector_struct.dot_prod_subscr_pos
attribution
...

Recognition in Java implementation

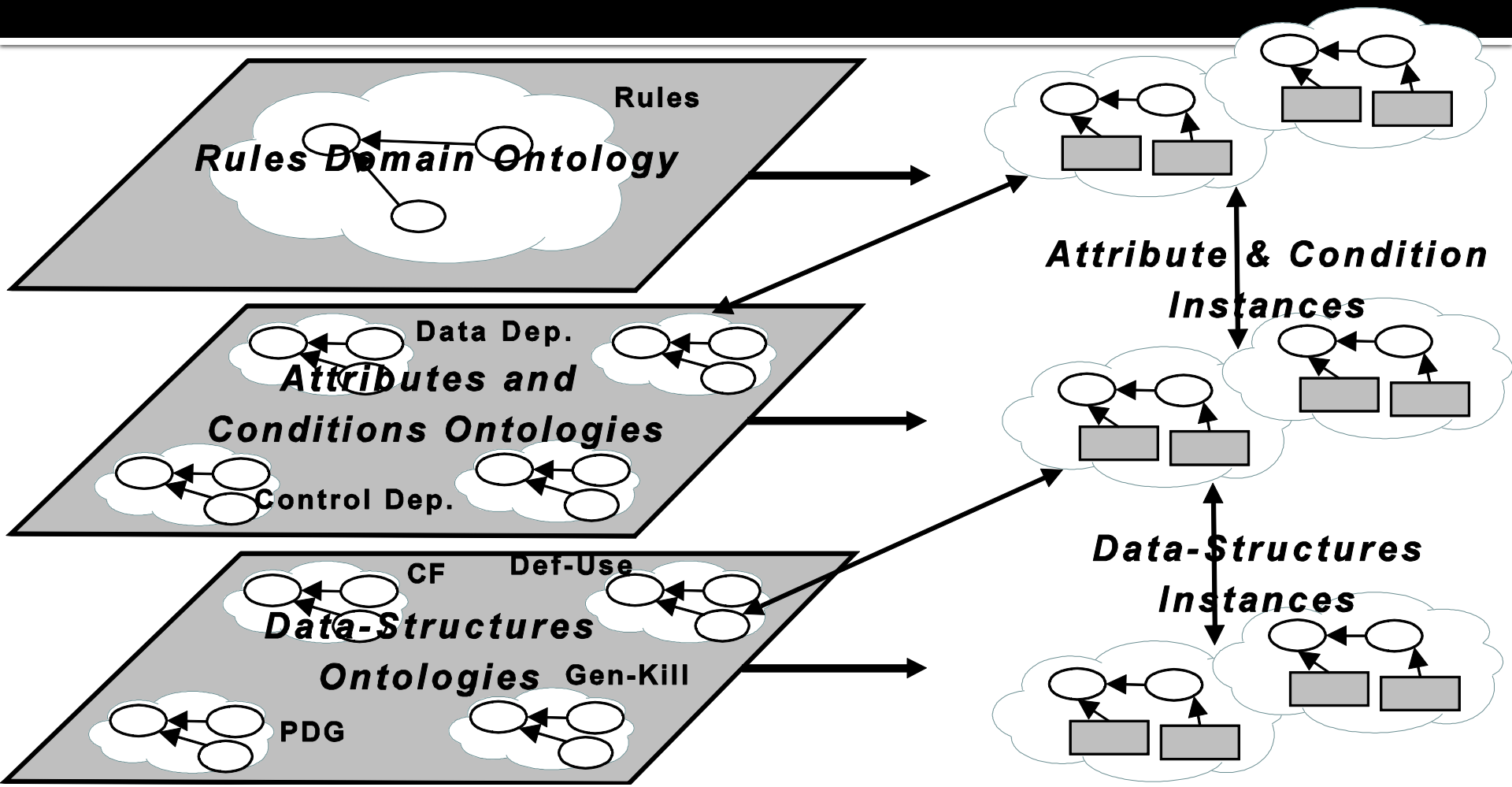


Variant: OWL based algorithmic patterns definition

Main advantages over a grammar specification:

- OWL has a clear syntax, and can be represented in graphical form. The graphical representation helps with readability, enhancing patterns definition validation.
- The OWL ability to be represented by directed graphs is usefull also in design phase, where users may specify their rules in a graphical way, without dealing with complex attribute grammars definitions.
- OWL is amenable to descriptive logics reasoning, and several reasoners exist (e.g. *Pellet*).
- Several clear and formal languages exist for querying information from reasoners (like as OWL-QL or SPARQL).
- Semantic annotation methods can be used to annotate source codes and related documentations (software assets).
- A good reasoning procedure may also lead to identification of patterns *similar* to the ones defined in the OWL knowledge-base, thus performing a *non exact* matching.

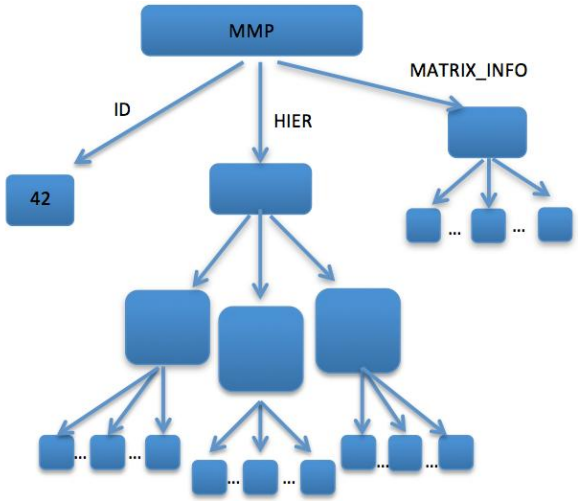
Ontology based multi-layer model for AP definition



Concept Recognition based Program Transformation

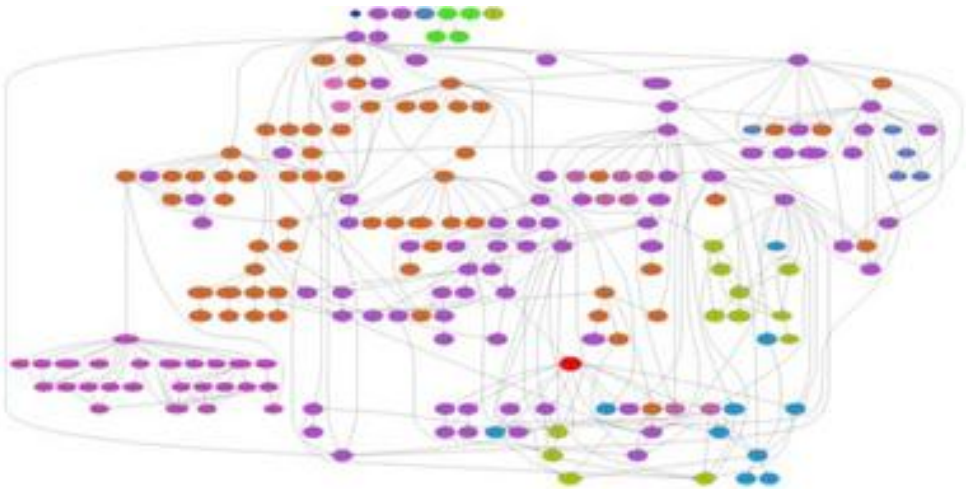
SUN Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

- The Algoritmic Concept recognizer produces representations of instances of retrieved algorithms (or algorithmic patterns) which are stored, including hierarchy of concepts composing the algorithm and their references to the source code.



Abstract Syntax Tree coloring

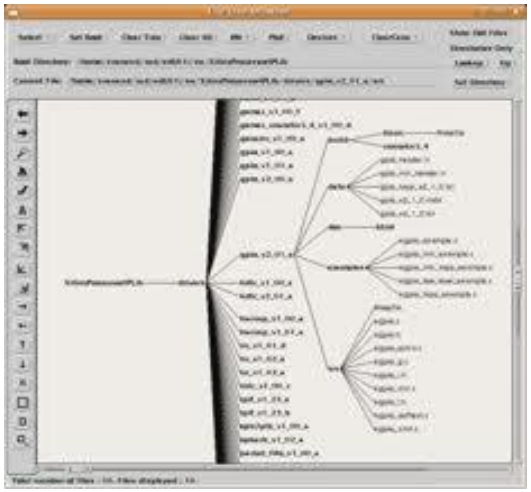
- By parsing the stored algorithmic instances, each node of the AST involved is "colored" with the instance of the algorithm it belongs to.
- Each node can belong to one or more concepts and more algoritms, so the coloring is indeed the adding of several attributes to nodes that reference an algorithm concepts instance list.



Source code transformation

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

- A transformation repository has been created. This contains, for each known algorithm and for each possible target architecture, a related set of transformations (or transformation sequences) optimized for that architecture.
- Several alternative or composable transformations can be driven by algorithmic recognition, ranging from unimodular loop transformations to pure native code generation or calls to optimized libraries.
- Solutions can also be selected depending on the problem size or runtime tuning parameters.



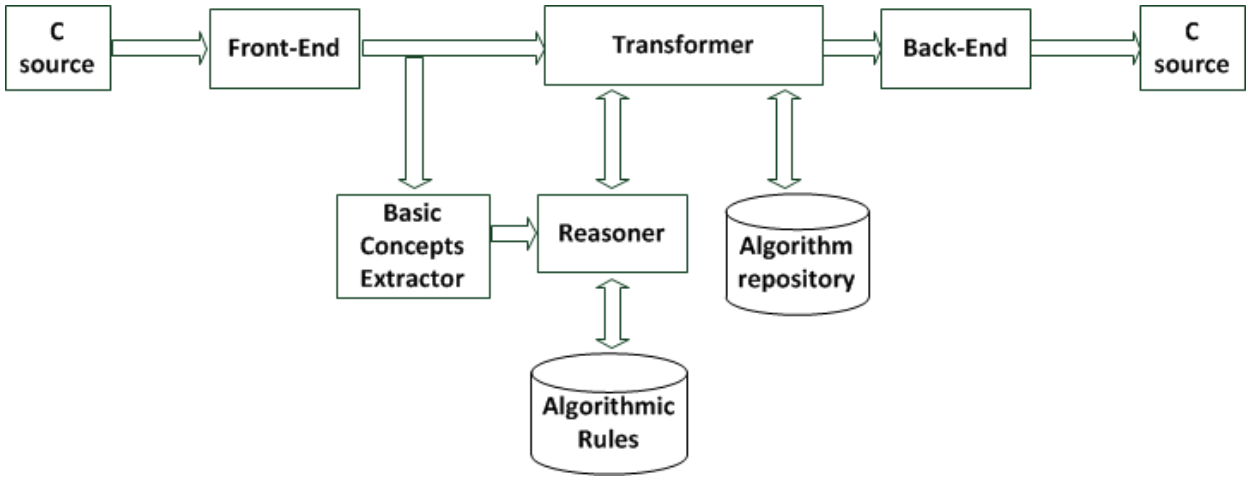
Source code transformation

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

- The source code of the program under investigation, is represented as a graph structure (Program dependence Graph); fragments of code mapping an algorithmic pattern are identified as a set of sub-graphs. The transformation rules applied on the graph model of the original source code: some sub-graphs are pruned, other modified, new one are inserted.
- The transformations are validated with dependence analysis.
- The result is a new graph that represents a new source code with the same semantics of the original one, but with different syntax suited for the target architecture. A final «unparse» operation produces, starting from the graph, new source code ready to be compiled on the target architecture.

Prototype targetting GPUs

- A prototype tool has been built with a parser to analyse source code written in C language, models the code and extracts the basic facts from it.
- A transformer with AST coloring functions and dependence analysis has been developed with Rose Compiler.
- Basic transformation rules have been defined to transform linear algebra code from sequential form to Nvidia CUDA library calls to take gain of GPUs.



Case study 1

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

- Factorization is allowed
- In the loop nest there are only statements belonging to the matrix matrix multiplication concept.
- The entire block can be replaced by CUBLAS call.

```
//matrix matrix product:
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {

            prod = 0;

            for (k = 0; k < n; k++)
            {
                prod += A[k * n + i] * B[j * n + k];
            }

            C[j * n + i] = alpha * prod + beta * C[j * n + i];
        }
    }
```

Case study 1 - result

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

```
//matrix matrix product:
/*Device matrix handlers */
  float *d_A;   float *d_B;     float *d_C;
  cublasStatus status;
/*CUBLAS inizialization*/
  status = cublasInit();
/*Device matrix Allocation*/
  status = cublasAlloc(nxa * nya,sizeof(float),((void **)&d_A));
  status = cublasAlloc(nxb * nyb,sizeof(float),((void **)&d_B));
  status = cublasAlloc(nxc * nyc,sizeof(float),((void **)&d_C));
/*Device matrix filled in with host data*/
  status = cublasSetVector(nxa * nya,sizeof(float),A,1,d_A,1);
  status = cublasSetVector(nxb * nyb,sizeof(float),B,1,d_B,1);
  status = cublasSetVector(nxc * nyc,sizeof(float),C,1,d_C,1);
/*CUBLAS matrix multiplication*/
  cublasSgemm('n','n',nxa,nyb,nya,alfa,A,nxa,B,nxb,beta,C,nxc);
/*Write results into Host*/
  status = cublasGetVector(nxc * nyc,sizeof(float),d_C,1,C,1);
/*Free Device memory*/
  status = cublasFree(d_A);
  status = cublasFree(d_B);
  status = cublasFree(d_C);
  cublasShutdown();
```

Case study 2

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

- Inside loop nest, there is a statement that doesn't belong to the concept.
- There is no data dependence.
- Factorization is allowed.

```
//matrix matrix product:
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            prod = 0;
            for (k = 0; k < n; k++)
            {
                prod += A[k * n + i] * B[j * n + k];
            }

            C[j * n + i] = alpha * prod + beta * C[j * n + i];

            // this statement should not be deleted
            D[j * n + i] = A[j * n + i];
        }
    }
```

Case study 2 - result

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

```
//matrix matrix product:
/*Device matrix handlers */
  float *d_A;   float *d_B;     float *d_C;
  cublasStatus status;
/*CUBLAS inizialization*/
  status = cublasInit();
/*Device matrix Allocation*/
  status = cublasAlloc(nxa * nya,sizeof(float),((void **)&d_A));
  status = cublasAlloc(nxb * nyb,sizeof(float),((void **)&d_B));
  status = cublasAlloc(nxc * nyc,sizeof(float),((void **)&d_C));
/*Device matrix filled in with host data*/
  status = cublasSetVector(nxa * nya,sizeof(float),A,1,d_A,1);
  status = cublasSetVector(nxb * nyb,sizeof(float),B,1,d_B,1);
  status = cublasSetVector(nxc * nyc,sizeof(float),C,1,d_C,1);
/*CUBLAS matrix multiplication*/
  cublasSgemm('n','n',nxa,nyb,nya,alfa,A,nxa,B,nxb,beta,C,nxc);
/*Write results into Host*/
  status = cublasGetVector(nxc * nyc,sizeof(float),d_C,1,C,1);
/*Free Device memory*/
  status = cublasFree(d_A);
  status = cublasFree(d_B);
  status = cublasFree(d_C);
  cublasShutdown();
  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
// this statement should not be deleted
      D[(j * n) + i] = A[(j * n) + i];
    }
  }
```

Case study 3

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

- Loop transformation not allowed
- Data dependence between a statement inside the loop not belonging to the concept and another that belongs to.

```
//matrix matrix product:
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            prod = 0;
            for (k = 0; k < n; k++)
            {
                prod += A[k * n + i] * B[j * n + k];
            }

            C[j * n + i] = alpha * prod + beta * C[j * n + i];

            // this statement does not allow code refactoring
            D[j * n + i] = prod;
        }
    }
```

Implementation details

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

- Searching for nodes belonging to the pattern.

```
void visitorTraversal::preOrderVisit(SgNode* n){

// looking for "forStatements"
    SgForStatement *forStatement = isSgForStatement(n);
    if ( forStatement != NULL )
    {

        bool forFound = false;
        //get the ast ID
        IDAttribute *idAttribute = ((IDAttribute *)n->getAttribute("ID"));
        ROSE_ASSERT(idAttribute != NULL);
        string myid = boost::lexical_cast<string>(idAttribute->getID());
        // check if the for id is into the prolog tree
        ptree::const_iterator it;
        for( it = getforStatementIdList(pt).begin(); it != getforStatementIdList(pt).end(); ++it){
            if (myid.compare(it->second.get_value<string>())==0){
                // the forStatement is part of the concept!
                forFound = true;
                break;
            }
        }
```

AST traversal

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

- Nested traversal

```
        if(forFound)
        {
            // adding the attribute to the ast node.
            ConceptAttribute *myConcept = new ConceptAttribute();
            string ID = pt->get("MMP.ID","");
            myConcept->addConcept(ID);
            forStatement->addNewAttribute((string)"Concept",(AstAttribute*)myConcept);

            // going inside the node looking for statements that are part of the concept
            nestedVisitorTraversal statement_traverse;
            statement_traverse.traverse(forStatement->get_loop_body(),preorder);
            trasformable = statement_traverse.isTransformable();
        }

        found = forFound;
```

Nested traversal

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

- Statements inside the loop nests

```
void nestedVisitorTraversal::visit(SgNode* n)
   {
     // getting the parent node
     SgNode* parent = n->get_parent();
     // casting on the parent (it could be a for statement)
     SgForStatement *forStatement = isSgForStatement(parent);
     // casting on the current node (it should be an exprStatement)
     SgExprStatement *exprStatement = isSgExprStatement(n);

     if(    (exprStatement != NULL) && // is an exprStatement
            ((forStatement == NULL) || (n == forStatement->get_loop_body()))&&
            (parent->variantT()!=V_SgForInitStatement)&& // The parent is not a ForInitStatement
            (transformable == true)
        )
        {
            bool stmFound = false;
            /*
            parte di controllo dell'ID del tutto equivalente al caso
            precedente
            */
            if(stmFound)
            {//...
```

AST node coloring

SUN Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

- … with data dependence checking

```
if(stmFound) {//...
    // adding the attribute
    string ID = pt->get("MMP.ID","");
    ROSE_ASSERT(ID.compare("") != 0 );
    ConceptAttribute *myNewConcept = new ConceptAttribute();
    myNewConcept->addConcept(ID);
    exprStatement->addNewAttribute((string)"Concept",(AstAttribute*)myNewConcept);
}
else
{
#if _DEBUG
printf ("This exprStatement is NOT part of mmp concept\n");
#endif

IDAttribute *idAttribute = ((IDAttribute *)n->getAttribute("ID"));
ROSE_ASSERT(idAttribute != NULL);
string myid = boost::lexical_cast<string>(idAttribute->getID());

transformable = !isDataDep(myid);
}
```

Transformation – insert header

- SgGlobal block query
- Header insertion

```
// add the include "cublas.h"
    nodeList = NodeQuery::querySubTree (project, V_SgGlobal);
    j = nodeList.begin();
    SgGlobal* globalScope = isSgGlobal(*j);

    if (globalScope != NULL)
    {
    SageInterface::insertHeader("cublas.h",PreprocessingInfo::before,globalScope);
    #if _DEBUG
    printf("...include added!!!\n");
    #endif
    }
```

Device data declaration

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

- Involved matrix and scalars declaration

```
// namespace SageBuilder for any build* function
varDec = buildVariableDeclaration("*d_"+n_A,buildFloatType()); //A
SageInterface::insertStatement(forStatement,varDec,true,true);

string comment = "\n/*Device matrix handlers */\n";
SageInterface::addTextForUnparser(varDec,comment,AstUnparseAttribute::e_before);

varDec = buildVariableDeclaration("*d_"+n_B,buildFloatType()); //B
SageInterface::insertStatement(forStatement,varDec,true,true);

varDec = buildVariableDeclaration("*d_"+n_C,buildFloatType()); //C
SageInterface::insertStatement(forStatement,varDec,true,true);

varDec = buildVariableDeclaration("status",buildOpaqueType("cublasStatus",myBlock)); //status
SageInterface::insertStatement(forStatement,varDec,true,true);
```

Device matrix allocation

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

```
        // adding the status = cublasAlloc(n,sizeof(float),(void**)d_A)
        assignStatement =   buildAssignStatement
                (
                    buildVarRefExp("status"),                       // status =
                    buildFunctionCallExp("cublasAlloc",             // cublasAlloc(
                            buildOpaqueType("cublasStatus",myBlock),
                            buildExprListExp
                                (
                                buildMultiplyOp(buildVarRefExp(n_aNx),buildVarRefExp(n_aNy)),
                                buildFunctionCallExp("sizeof",      // sizeof(float),
                                                buildIntType(),
                                                buildExprListExp(buildVarRefExp("float")),
                                                myBlock
                                                ),
                                buildCastExp(                       //(void**)d_A)
                                                buildVarRefExp("&d_"+n_A),
                                                buildPointerType(buildPointerType(buildVoidType())),
                                                SgCastExp::e_C_style_cast
                                            )

                                ),
                            myBlock)
        );
        SageInterface::insertStatement(forStatement,assignStatement,true,true);

        comment = "\n/*Device matrix Allocation*/\n";
    SageInterface::addTextForUnparser(assignStatement,comment,AstUnparseAttribute::e_before);
```

Data transfer host->device

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

```
// adding the status = cublasSetVector(n2,sizeof(float),A,1,d_A,1)
assignStatement =   buildAssignStatement(
buildVarRefExp("status"),
buildFunctionCallExp("cublasSetVector",
                     buildOpaqueType("cublasStatus",myBlock),
                     buildExprListExp(
                         buildMultiplyOp(buildVarRefExp(n_aNx),buildVarRefExp(n_aNy)),
                         buildFunctionCallExp("sizeof",
                                              buildIntType(),
                                              buildExprListExp(buildVarRefExp("float")),
                                              myBlock
                                              ),
                         buildVarRefExp(n_A),
                         buildIntVal(1),
                         buildVarRefExp("d_"+n_A),
                         buildIntVal(1)
                         ),
                     myBlock)
             );
SageInterface::insertStatement(forStatement,assignStatement,true,true);
```

CUBLAS MatrixMatrix Multiplication

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

```
SgExprListExp* listExp =
        buildExprListExp(   buildCharVal('n'),buildCharVal('n'),    //n carattere
                            buildVarRefExp(n_aNx),  //dim
                            buildVarRefExp(n_bNy),  //dim
                            buildVarRefExp(n_aNy),  //dim
                            buildVarRefExp(n_alfa), //alfa
                            buildVarRefExp(n_A),    //A
                            buildVarRefExp(n_aNx),  //leading dimension (righe)
                            buildVarRefExp(n_B),    //B
                            buildVarRefExp(n_bNx)   //leading dimension (righe)
                            );
listExp->append_expression(buildVarRefExp(n_beta)); //beta
listExp->append_expression(buildVarRefExp(n_C));    //C
listExp->append_expression(buildVarRefExp(n_cNx));  //leading dimension (righe)

SgExprStatement* callStmt = buildFunctionCallStmt   (cublasSgemm,buildVoidType(),listExp,basicBlock);
SageInterface::insertStatement(forStatement,callStmt_1,true,true);

comment = "\n/*CUBLAS matrix multiplication*/\n";
SageInterface::addTextForUnparser(callStmt,comment,AstUnparseAttribute::e_before);
```

Loop body removal

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

```
2   // query the ast looking for "forStatements"
3           Rose_STL_Container<SgNode*> forStatementList;
4           Rose_STL_Container<SgNode*>::iterator j;
5           nodeList = NodeQuery::querySubTree (project, V_SgForStatement);
6
7   // nested query, looking for "coloured forStatements"
8           forStatementList = NodeQuery::queryNodeList (nodeList,&querySolverStatement);
9           j = forStatementList.end();
10          while( j != forStatementList.begin())
11          {
12              j--;
13              SgForStatement* forStatement = isSgForStatement(*j);
14              ROSE_ASSERT(forStatement != NULL);
15              visitorTraversalRem myTraversalrem;
16              // Call the traversal starting at the forStatement node of the AST
17              myTraversalrem.traverse((SgNode*)forStatement);
18
19              // removing the statement using the middle level
20              // it deletes the comments too
21              if ( myTraversalrem.is_removable() ) MiddleLevelRewrite::remove(forStatement);
22          }
```

Recursive application

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

```
NodeQuerySynthesizedAttributeType querySolverStatement (SgNode * astNode)
    {
        // casting on the current node, if should be a Statement
        SgStatement* statement = isSgStatement(astNode);
        if (statement != NULL){
        // getting the attribute
        ConceptAttribute *myConcept = (ConceptAttribute*)statement->getAttribute("Concept");
        if (myConcept != NULL)
        {
            // checking the concept ID
            string ID = myptree->get("MMP.ID","");
            ROSE_ASSERT(ID.compare("") != 0 );
            if (myConcept->hasConcept(ID)) {
                // the node has the correct concept attribute, so is added to the output
                returnNodeList.push_back (astNode);
            }
        }

     }
     return returnNodeList;
}
```

Ongoing and future work

SUN
Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

- Extend the set of recognized concepts by adding more rules (beyond linear algebra).
- Add runtime analysis probes in order to adopt alternative codes depending on the size of the problem.
- Investigate on code that make use of dynamic memory allocation and pointer.
- Add support for OpenCL for heterogeneous architectures.
- Develop a GUI to support user driven transformations

Questions?

SUN Seconda Università degli Studi di Napoli
DIPARTIMENTO di INGEGNERIA dell'INFORMAZIONE

- Beniamino Di Martino – beniamino.dimartino@unina.it
- Pasquale Cantiello – pasquale.cantiello@unina2.it