Semantic-based Service trading: Application to Linear Algebra[⋆⋆⋆]

Michel Daydé, Aurélie Hurault, Marc Pantel

IRIT - ENSEEIHT, 2 rue Camichel, B.P. 7122, F-31071 TOULOUSE CEDEX 7
{Aurelie.Hurault,Marc.Pantel}@enseeiht.fr

Abstract. One of the great benefit of computational grids is to provide access to a wide range of scientific software and computers with different architectures. It is then possible to use a variety of tools for solving the same problem and even to combine these tools in order to obtain the best solution technique.
Grid service trading (searching for the best combination of software and execution platform according to the user requirements) is thus a crucial issue. Trading relies both on the description of available services and computers, on the current state of the grid, and on the user requirements. Given the large amount of services available on the Grid, this description cannot be reduced to a simple service name.
We present in this paper a more sophisticated service description similar to algebraic data type. We then illustrate how it can be used to determine the combinations of services that answer a user request. As a side effect, users do not make direct explicit calls to grid-services but talk to a more applicative-domain specific service trader.
We illustrate this approach and its possible limitations within the framework of dense linear algebra. More precisely we focus on Level 3 BLAS ([DDDH90a,DDDH90b]) and LAPACK ([ABB$^+$99]) type of basic operations.

1 Introduction

Given all the services deployed on a grid, finding the most appropriate service or composition of services which are able to fulfill a user request is quite challenging and requires more than the knowledge of the service's signatures.

We introduce here an approach that consists in adding additional semantic information to the services in order to reduce ambiguity in their description and allow to find the services or combination of services that provide good answers to a user request using equational unification to identify all the possible choices. As a benefit, users do not need to make explicit call to specific services over the grid (such as some GridRPC call for example). The user does not need to

[⋆] This work has been partially supported by the French Ministery of Research throught the GRID-TLSE Project from ACI «*Globalisation des Ressources Informatiques et des Données*»

[⋆⋆] Candidate to the Best Student Paper Award know the exact name of the service he is looking for, he just has to describe the mathematical operation he wants to compute in a given applicative domain. Our service trader finds the appropriate service or combination of services (eventually it can provide the user a list of possible choices and ask him to choose the best one given the mathematical operation and not the library name). The interaction with the middleware can then be hidden behind a domain specific interface.

We take examples from dense linear algebra for the sake of simplicity, but this approach can be extended to other areas since the algorithm is generic and parameterized by the description of the application domain.

2 Problem description

A key issue in advanced trading of services is the choice of a description formalism for the available services. The comparison between the available services and the user's requests depends on the formalism chosen for this description.

2.1 Different approaches

The simplest description used in most SOA (Service Oriented Architecture) such as RPC, CORBA, COM, DCOM, RMI makes only use of the service signatures (input and output types of parameters). This information has the advantage to be easily available. But it is not sufficient for sophisticated trading, even if we use type isomorphisms to remove the problems of parameter position. Indeed, with such an approach there is no way to distinguish addition from multiplication as both share the same signature.

We can add keywords or meta-data to the service signature (this is currently the case in the GRID-TLSE project [PPA05,Pan04]). This formalism allows an easy comparison of the services and the request. But this description requires a preliminary agreement to define the keywords and their meaning, with all the ambiguities implied by the natural language. Another disadvantage is the difficulty to describe a complex service. How to describe without ambiguity and with keywords some Level 3 BLAS procedures such as $SGEMM$ expressed by the following formula: $\alpha * \mathbf{A} * \mathbf{B} + \beta * \mathbf{C}$?

Another approach which extends keywords and metadata is based on ontologies such as OWL. The advantage of ontologies is the possibility to have a formal description. It also provides the logic associated to reason about the descriptions. The disadvantage is that we do not control this logic which can be undecidable. The ontologies also need a preliminary agreement to define the keywords and their meaning. In the case of ontologies, this preliminary agreement is formally described thanks to relation between the different keywords, this is a main advantage over the previous approach. Moreover the definition of an ontology is not trivial and hard to achieve for a non specialist.

In the Monet[1] and HELM[2] projects the description of the computational services is based on MathML[3] and OpenMath[4] which provide an accurate description. But the comparison of services is based on RDF and ontologies which did not allow easily to adapt and combine services during the trading.

We follow the same approach as the NASA Amphion project [SWL+94] and more particularly the theorem prover SNARK (independence of the application domain, reasoning based on starting from a description of the domain). But, this project relies on *«term rewriting and the paramodulation rule for reasoning about equality»*. This supposes that *«a recursive path ordering is supplied when the application domain theory is formulated»*. The last constraint require that the user is familiar with complex rewriting technics. One of our main requirements is that the user should not need to know anything about the underlying technologies.

We are looking for a simpler description, with the least possible ambiguities, that can be specified by a specialist of a given domain without the help of a specialist on ontologies or the use of complex knowledge in rewriting techniques.

For all theses reasons, we have opted for a description similar to algebraic data types. The advantages of this description is the possibility of describing without ambiguity both the services and the knowledge of the main properties of the domain that are required for composing services to fulfill the user requests.

We describe in more details our semantic-based description of services in the next section. The trading algorithm is described in Section 3. Examples and possible limitations of this approach when looking for the best combination of services are reported in Section 4. We finally conclude in Section 5.

2.2 An algebraic data type based description for advanced trading

As said before, the semantic used is similar to algebraic data type description [GH78]. Indeed the required information are:

- the types (or sorts) used;
- the main operators of the specific domain and their signatures (we allow overloading);
- the operators properties (such as commutativity and associativity) and the equations that link operators.

When considering dense linear algebra and basic operations such as BLAS and LAPACK, we define:

- Types: $Int, Real, Char, Matrix, \ldots$
- Operators and their signatures:
  - Addition of matrices: $+ : Matrix \times Matrix \rightarrow Matrix$

[1] http://monet.nag.co.uk/cocoon/monet/index.html
[2] http://helm.cs.unibo.it/
[3] http://www.w3.org/Math/
[4] http://www.openmath.org/cocoon/openmath/index.html

    - Multiplication of a matrix by a scalar: $* : Real \times Matrix \rightarrow Matrix$
    - Matrix multiplication: $* : Matrix \times Matrix \rightarrow Matrix$
    - Transpose of a matrix: $T : Matrix \rightarrow Matrix$
    - Identity: $I :\rightarrow Matrix$
    - Null matrix: $O :\rightarrow Matrix$
    - ...
  - Properties:
    - Addition $+$: commutative and associative (can be expressed directly by the corresponding equations)
    - Multiplication $*$: associative (can be expressed directly by the corresponding equations)
    - Neutral element I: $a : Matrix \quad I * a = a$
    - Absorbant element O: $a : Matrix \quad O * a = O$
    - Distributivity $*/+$:
      $a : Matrix\ b : Matrix\ c : Matrix \quad a * (b + c) = (a * b) + (a * c)$
    - Distributivity $*/+$:
      $a : Real\ b : Matrix\ c : Matrix \quad a * (b + c) = (a * b) + (a * c)$
    - ...

The last two equations can be factorized by:
$a :\ \ b : Matrix\ c : Matrix \quad a * (b + c) = (a * b) + (a * c)$.
That means that the equation is valid for all the types of a for which $a * (b + c)$ and $(a * b) + (a * c)$ are well typed.

With this description, we can describe some of the Level 3 BLAS procedures in a formalism very similar to the official BLAS specification [DDDH90a].

SGEMM performs one of the matrix-matrix operations:

$$\mathbf{C}{=}\alpha{*}\mathrm{op}(\mathbf{A}){*}\mathrm{op}(\mathbf{B}){+}\beta{*}\mathbf{C}$$

where α and β are scalars, op($\mathbf{A}$) and op($\mathbf{B}$) are rectangular matrices of dimensions m$\times$k and k$\times$n, respectively, $\mathbf{C}$ is a m $\times$ n matrix, and op($\mathbf{A}$) is $\mathbf{A}$ or $\mathbf{A}^T$.

In the trader, *SGEMM* will be described by an XML document whose meaning is:

```
SGEMM(TRANSA:Char, TRANSB:Char, M:Int, N:Int, K:Int, ALPHA:Real,
      A:Matrix, LDA:Int, B:Matrix, LDB:Int, BETA:Real, C:Matrix, LDC:Int)
C <- ALPHA * op(TRANSA,A) * op(TRANSB,B) + BETA * C
```

Among the equations of the domain, will be: $op('n', a) = a$ and $op('t', a) = a^T$.

However, this description is not rich enough for sophisticated trading involving service combination. Some numerical properties of the matrix are very important to select a suitable Level 3 BLAS procedure. For example when considering matrix-matrix multiplication, symmetry of one of the matrices involved in the operation may lead to select $SSYMM$ rather than $SGEMM$ and similarly when dealing with a triangular matrix that is supported by $STRMM$. To take into account these properties, subtypes have been introduced in the description. Some restrictions are required about the definitions of subtypes. The relation on types must be a partial order relation (antisymmetric, transitive and reflexive), and must verify some constraints expressed in [CGL92].

To the previous description, we add:

- Types:
  - Invertible matrices: $InvMatrix < Matrix$
  - Symmetric matrices: $SymetricMatrix < Matrix$
  - Triangular matrices: $TriangularMatrix < Matrix$
  - Invertible triangular matrices:
    $InvTriangularMatrix < TriangularMatrix$,
    $InvTriangularMatrix < InvMatrix$
  - ...
- Operators and their signatures (we can specify the conservation of a property by an operator):
  - Multiplication of a symmetric matrix by a scalar:
    $* : Real \times SymetricMatrix \rightarrow SymetricMatrix$
    (the symmetric property is conserved)
  - Multiplication of a triangular matrix by a scalar:
    $* : Real \times TriangularMatrix \rightarrow TriangularMatrix$
  - Multiplication of an invertible triangular matrix by a non-zero scalar:
    $* : NzReal \times InvTriangularMatrix \rightarrow InvTriangularMatrix$
  - Transpose of a triangular matrix:
    $T : TriangularMatrix \rightarrow TriangularMatrix$
  - ...
- ...

In the examples, we give high level properties, but we can enrich the description to specify more precisely the matrix. The user which defines the application domain chosses the level of granularity of the description. It is important to notice that the impact of a more precise description is in relation with the new equations that the new properties may imply. Adding types is not very costly but it generally leads to introduce new equations which is more expensive.

We are now able to define all the services.

$SSYMM$ performs one of the matrix-matrix operations:

$$\mathbf{C}=\alpha*\mathbf{A}*\mathbf{B}+\beta*\mathbf{C}, \text{ or } \mathbf{C}=\alpha*\mathbf{B}*\mathbf{A}+\beta*\mathbf{C}$$

where α and β are scalars, **A** is an m $\times$ m symmetric matrix (only the upper or lower triangular part is used), **B** and **C** are m $\times$ n matrices.

In the trader $SSYMM$ will be described by an XML document whose meaning is:

```
SSYMM(SIDE:Char, UPLO:Char, M:Int, N:Int, ALPHA:Real, A:SymetricMatrix,
      LDA:Int, B:Matrix, LDB:Int, BETA:Real, C:Matrix, LDC:Int)
IF SIDE='l' THEN C <- ALPHA * A * B + BETA * C
IF SIDE='r' THEN C <- ALPHA * B * A + BETA * C
```

In practice the description is not exactly this one to take into account the *UPLO* parameter. This point will be discussed later.

STRSM solves one of the matrix equations:

$$\mathbf{A}*\mathbf{X}=\alpha*\mathbf{B},\ \mathbf{A}^{\mathbf{T}}*\mathbf{X}=\alpha*\mathbf{B},\ \mathbf{X}*\mathbf{A}=\alpha*\mathbf{B},\text{ or }\mathbf{X}*\mathbf{A}^{\mathbf{T}}=\alpha*\mathbf{B}$$

where α is a scalar, **X** and **B** are m $\times$ n matrices and **A** is a unit, or non-unit, upper or lower triangular matrix. **B** is overwritten by **X**.

In the trader *STRSM* will be described by an XML document whose meaning is:

```
STRSM(SIDE:Char, UPLO:Char, TRANSA:Char, DIAG:Char, M:Int, N:Int,
      ALPHA:Real, A:InvTriangularMatrix, LDA:Int, B:Matrix, LDB:Int)
IF SIDE='l' THEN B <- ALPHA * op(TRANS,A^{-1}) * B
IF SIDE='r' THEN B <- ALPHA * B * op(TRANS,A^{-1})
```

In practice the description is not exactly this one to take into account the *UPLO* and *DIAG* parameters.

The matrix **A** is not necessary a triangular matrix, but can be considered as a triangular matrix (*UPLO* indicates if it is a lower or upper triangular matrix and *DIAG* if it is a unit matrix). This is the case when this matrix is used to store two different triangular matrices (like after a LU factorization). This problem needs more work to reach an acceptable treatment.

Currently, *STRSM* is defined with **A** not necessary a triangular matrix, and with operation done on the upper or lower part, but it is not a good solution because the real problem is not *STRSM* but the object which represents several objects. We must design a general solution for this problem instead of the ad-hoc approach currently in use which have an impact on all the services.

These descriptions illustrate that we can manage parameters which are both input and output. We can also specify the service in function of a given parameter.

We can now describe the services and the user's request. Our aim is to find the services or the combination of services that satisfies the client's request. For doing so, we first compute all the available services and combination of available services which answer the user request. Then, in a second step, we will chose the «best» one, according to the user's criteria. We may combine these two step for a better effectiveness.

3 Computing the combination of services corresponding to an user's request.

To identify all the services and combinations of services that answer the user's problem, we compare the description of the user's problem with the description of all the services, taking into account the properties of the domain (here dense linear algebra).

3.1 The trading algorithm

Our comparison of two descriptions is based on equational unification [BS01] and in particular on the set of transformations of Gallier and Snyder which has been proved to be sound and complete [GS89]. This system has been adapted to add types and subtypes and also to improve the performance. The problems introduced by the overloaded functions with subtyping are treated as in [CGL92].

To control the algorithm we use two parameters: the depth of combination allowed and the number of equations applied. This second number is really critical because our algorithm has an exponential complexity for this parameter. Further improvements to our algorithm are required in the future to limit the complexity of computing the combinations of services corresponding to a request. It may be interesting to use ad-hoc treatments for properties such as commutativity, associativity, distributivity, zero element, identity element, ... The general principle of the algorithm is explained in details in [HP06].

3.2 Examples

We consider examples arising in dense linear algebra with a complete description of this domain.

For all the following examples, the results given, are some among all the results computed by the trader. The number of equations allowed and the depth of combination given are the minimum ones. If more equations are allowed to be applied and a bigger depth of combination is allowed, the number of results will grow.

Example 1 The available services are the ones from the Level 3 BLAS. The request of the user is $A : Matrix,\ B : Matrix,\ C : Matrix \quad C = A * B * C$.

One combination of services computed by the trader is:

```
Matrix p2=Any x1;
SGEMM('n','n',m?,n?,k?,1.,B,lda?,C,ldb?,0.,p2,ldc?); \\p2<-B*C
Matrix p1=Any x1;
SGEMM('n','n',m?,n?,k?,1.,A,lda?,p2,ldb?,0.,p1,ldc); \\p1<-A*p2
p1;
```

where *Any* $x1$ can be any matrix and the parameters following by a "?" are the ones we cannot determine, they will be determined later on.

To find this solution, the trader must be run with more than 5 equations allowed to be applied and a depth of combination allowed of at least 1.

Example 2 Now, the available services are the ones from the Level 3 BLAS and some from LAPACK [ABB+99] (row interchanges *SLASWP*, the Cholesky factorization *SPOTRF* and the LU factorization *SGETRF*). The user wants to solve the linear system with multiple right-hand side members $Ax = B$ (where no property is known about A). One answer computed by the trader is:

```
InvMatrix p2=A;
Vector p6=ipiv?;
SGETRF(m?,n?,p2,lda?,p6,info?); \\p2<-fatorization LU of A (A= P*L*U)
Matrix p5=B;
SLASWP(n?,p5,lda?,k1?,k2?,p6,incx?); \\p5<-row interchanges of B
Matrix p3=p5;
STRSM('l','l','n',u?,m?,n?,1.,p2,lda?,p3,ldb?); \\solve L*x=p5; p3<-x;
Matrix p1=p3;
STRSM('l','u','n',u?,m?,n?,1.,p2,lda?,p1,ldb?); \\solve U*x=p3; p1<-x;
p1;
```

To find this solution, the trader must be run with more than 7 equations allowed to be applied and a depth of combination allowed of at least 3.

Example 3 The example in similar conditions as the previous one but now **A** is a symmetric positive definite matrix.

The trader computes the following compositions of services:

```
SymDefPosMatrix p2=A:SymDefPosMatrix ;
Vector p6=ipiv?;
SGETRF(m?,n?,p2,lda?,p6,info?); \\ p2<-fatorization LU of A (A= P*L*U)
Matrix p5=B;
SLASWP(n?,p5,lda?,k1?,k2?,p6,incx?); \\ p5<-row interchanges of B
Matrix p3=p5;
STRSM('l','l','n',diag?,m?,n?,1.,p2,lda?,p3,ldb?); \\solve L*x=p5; p3<-x;
Matrix p1=p3;
STRSM('l','u','n',diag?,m?,n?,1.,p2,lda?,p1,ldb?); \\solve U*x=p3; p1<-x;
p1;
```

To find this solution, the trader must be run with more than 7 equations allowed to be applied and a depth of combination allowed of at least 3.

and

```
SymDefPosMatrix p2=A;
SPOTRF('u',p2,info); \\ p2<- Cholesky factorization of A (A=U{^T}*U)
```

```
Matrix p3=B;
STRSM('l','u','t',diag?,m?,n?,1.,p2,lda?,p3,ldb?); \\solve U{^T}*x=B; p3<-x;
Matrix p1=p3;
STRSM('l','u','n',diag?,m?,n?,1.,p2,lda?,p1,ldb?); \\solve U*x=p3; p1<-x;
p1;
```

To find this solution, the trader must be run with more than 6 equations allowed to be applied and a depth of combination allowed of at least 3.

and

```
SymDefPosMatrix p2=A;
SPOTRF('l',p2,info); \\p2<- Cholesky factorization of A (A=L*L{^T})
Matrix p3=B;
STRSM('l','l','n',diag?,m?,n?,1.,p2,lda?,p3,ldb?); \\solve L*x=B; p3<-x;
Matrix p1=p3;
STRSM('l','l','t',diag?,m?,n?,1.,p2,lda?,p1,ldb?); \\solve U{^T}*x=p3; p1<-x;
p1;
```

To find this solution, the trader must be run with more than 7 equations allowed to be applied and a depth of combination allowed of at least 3.

The first solution is the same as in the general case for **A** (i.e. **A** general square). The other uses the fact that **A** is positive definite and replaces the LU factorization by a Cholesky factorization which is a better solution.

Example 4 The example in similar conditions as the previous ones but now **A** is an invertible upper triangular matrix.

The following solution is found:

```
Matrix p1=B;
STRSM('l','u','n',diag?,m?,n?,1.,A,lda?,p1,ldb?); \\solve A*x=B; p1<-x;
p1;
```

To find this solution, the trader must be run with more than one equation allowed to be applied and any depth of combination (since no combination is needed).

These examples illustrate the fact that the trader look for several solutions taking into account the properties of the domain and of the parameters. All the solutions do not have the same quality, a choice must be made among these solutions.

4 Choosing the solution to be run

The trading algorithm finds all the suitable solutions within given depth and number of equations applied. We still have to select the one that will be executed. Among the set of solutions produced, only the most relevant ones are kept. When this first choice is made, we will interact with a grid middleware to finally select the one to execute.

4.1 Discarding solutions without interest

When looking for a solution, we compare the request with all the services. For a given service, we may find a solution that is a combination of services involving subproblems to be solved. In this case, we run again the algorithm on the subproblems. To avoid computation of uninteresting solutions, we do not run again the algorithm if there is a subproblem which is the same as the initial problem.

Example: We want to compute $a + b$ and we have the service $x * y$. Then, $\{x \leftarrow a + b,\ y \leftarrow I\}$ is a solution requiring a combination, but we discard it.

We also simplify the request before running again the algorithm. This is necessary, to avoid running the algorithm on requests such that $a + O$, $a * I$, ...

4.2 Selecting the most relevant solutions

Piloting research To improve the search of relevant results, we can explore first the most interesting services.

To decide whether a service is interesting, we consider its complexity (static information) and its availability (dynamic information). By exploring these services at the beginning of the trading process, the initial solutions found will be the most relevant ones, since they will be the least complex and they will be available.

Static information are not sufficient since we are on a Grid whose QoS can change dramatically, and we must take into account the network load, the data migration, ... Indeed, we prefer to satisfy a request with a service located on a server which has a strong availability rather than with a service located on a busy server assuming that both servers have the same performance. Improving the computation of services using these dynamic informations that can be provided by a middleware such as Diet (see section 4.3), used in the GRID-TLSE project, may be crucial for performance and will require further improvements and experiments in the trading algorithm .

Another way to find first the most relevant solutions is to change the way we traverse the research tree. Currently, we do a breadth first traversal. It may be interesting to use a more complex traversal based on a weighting of the branches. This weight will be calculated in function of the complexity of the subproblem.

Sorting results The obtained results must be sorted. Currently, this sort is done by considering the complexity of the services. Services which have the same complexity, are sorted in function of their parameters.

Assume that $f(x, y, x, O)$, $f(x, y, O, O)$ and $f(x, y, Any, O)$ solve the problem. The most interesting result is the last one ($f(x, y, Any, O)$) because it is the most general. $f(x, y, O, O)$ is more interesting than $f(x, y, x, O)$, because the null matrix is, in general, less complex than the user matrix. In the general case, services with same complexity will be sorted according to the increasing numbers of *Any*, constants and parameters given by the user within their parameters.

4.3 Interaction with a middleware

The trader can then choose to transmit the most relevant result to the middleware which will schedule the chosen composite service. It can also choose to transmit several relevant results. The choice among the different results will be done by the middleware. Several environments provide the features needed: NetSolve [AAB+01], NINF [TNS+03], DIET [DIE], NEOS [NEO], or RCS [AGM97]. DIET is the middleware used in the GRID-TLSE project, where our work takes place.

In the case of simple service (without combination), the state of the machine where the service is located, its capacity, its availability, ... will be considered. In the case of combination of services, in addition to these information, the data dependencies must also be taken into account to evaluate the costs in term of communication between the computers running the different services. Indeed, the local execution (even on a less powerful server) might be quicker than the remote execution because of the extra overhead due to data movements.

If none of the services is satisfying to the middleware, it can ask for more results until it obtains satisfaction. More complex searches may then be started. As soon as the middleware obtains a valid solution, it executes the request (or the sequence of requests).

5 Conclusion

We have described an approach for advanced trading of services based on an algebraic data type like description of applicative domain and services. Our trading algorithm allows to compose existing services in order to satisfy the user request.

The trading algorithm first computes all the possible solutions within a given depth and a given number of equations examined. The main difficulty in that process is to limit the exponential complexity of the search for solutions by discarding the less relevant ones. Some issues are currently explored consisting in using a different strategy for searching in the solution tree: aiming at decreasing the number of branches explored, use of a cache mechanism for avoiding recomputing solutions, ... Finally, within the set of solutions computed, a selection is made by considering the complexity of the operations and their parameters.

The current trading algorithm provides the appropriate results but it is still very preliminary and further improvements on time and memory performances are required. Our goal would be to incorporate such a trading mechanisms within interactive scientific computing environments such as MATLAB or SciLAB to allow users to take advantage of grid services - when adequate - in a transparent way (without explicit calls) and to interact with a middleware to benefit of their scheduling capacity.

References

[AAB$^+$01] D. Arnold, S. Agrawal, S. Blackford, J. Dongarra, M. Miller, K. Sagi, Z. Shi, and S. Vadhiyar. Users' Guide to NetSolve V1.4. Computer Science Dept. Technical Report CS-01-467, University of Tennessee, Knoxville, TN, July 2001.

[ABB$^+$99] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, Jack J. Dongarra, J. Du Croz, S. Hammarling, A. Greenbaum, A. McKenney, and D. Sorensen. *LAPACK Users' guide (third ed.).* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

[AGM97] P. Arbenz, W. Gander, and J. Mori. The Remote Computational System. *Parallel Computing*, 23(10):1421–1428, 1997.

[BS01] F. Baader and W. Snyder. Unification theory. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 8, pages 445–532. Elsevier Science, 2001.

[CGL92] Giuseppe Castagna, Giorgio Ghelli, and Giuseppe Longo. A calculus for overloaded functions with subtyping. In *Proceedings of the ACM Conference on Lisp and Functional Programming*, volume 5, pages 182–192, 1992.

[DDDH90a] J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling. Algorithm 679. a set of Level 3 Basic Linear Algebra S ubprograms. *ACM Transactions on Mathematical Software*, 16:1–17, 1990.

[DDDH90b] J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling. Algorithm 679. a set of level 3 basic linear algebra subprograms: model implementation and test programs. *ACM Transactions on Mathematical Software*, 16:18–28, 1990.

[DIE] DIET. http://graal.ens-lyon.fr/DIET.

[GH78] John V. Guttag and James J. Horning. The algebraic specification of abstract data types. *Acta Inf.*, 10:27–52, 1978.

[GS89] J. H. Gallier and W. Snyder. Complete Sets of Transformations for General E-Unification. *Theor. Comput. Sci.*, 67(2-3):203–260, 1989.

[HP06] Aurélie Hurault and Marc Pantel. Mathematical service trading based on equational matching. In *Proceedings of the 12th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning (Calculemus 2005)*, volume 151, pages 161–177. Electronic Notes in Theoretical Computer Science, 21 March 2006.

[NEO] NEOS - Server for Optimization. http://www-neos.mcs.anl.gov/neos/.

[Pan04] M. Pantel. Test of Large Systems of Equations on the Grid: Meta-Data for Matrices, Computers, and Solvers. In *PMAA'04*, 2004.

[PPA05] Marc Pantel, Chiara Puglisi, and Patrick Amestoy. Grid, Components and Scientific computing. In *Submission to Euro-Par 2005*, 2005.

[SWL$^+$94] Mark E. Stickel, Richard J. Waldinger, Michael R. Lowry, Thomas Pressburger, and Ian Underwood. Deductive composition of astronomical software from subroutine libraries. In *CADE*, pages 341–355, 1994.

[TNS$^+$03] Yoshio Tanaka, Hidemoto Nakada, Satoshi Sekiguchi, Toyotaro Suzumura, and Satoshi Matsuoka. Ninf-G: A Reference Implementation of RPC-based Programming Middleware for Grid Computing. *Journal of Grid Computing*, 1(1):41–51, 2003.