Courtage sémantique de services de calcul

Sémantique explicite pour inférence de services

Aurélie Hurault Marc Pantel

Description des services

Plusieurs niveaux de description des services :

- signature (nom et types) — niveau ontologique usuel ;
- **sémantique** (spécification de ce que le service fait) — spécification fonctionnelle ;
- comportement (protocoles d'interaction) — spécification comportementale ;
- QoS (propriétés non fonctionnelle).

Découverte de services basée sur la sémantique

- Étapes :
  1. Spécialiste : décrit la fonctionnalité du service.
  2. Utilisateur : décrit la fonctionnalité recherchés.
  3. Algorithme : compare les descriptions pour trouver les services et compositions de services qui résolvent le problème.
- Descriptions possibles au niveau sémantique :
  - Nom des services, signatures, mot clés et/ou ontologies
  - Limites :
    - Nom : nécessité de connaître la nomenclature
      BLAS : SGEMM, E04-NAG : E04ABF
    - Signature : pas assez précis
      addition and multiplication : Matrix $\times$ Matrix $\rightarrow$ Matrix
    - Mot clés : ambiguë
      BLAS, SGEMM : $\alpha * A * B + \beta * C$
    - Ontologies :
      Pas de contrôle du solveur.

Découverte de services basée sur la sémantique

- Étapes :
  1. Spécialiste : décrit la fonctionnalité du service.
  2. Utilisateur : décrit la fonctionnalité recherchés.
  3. Algorithme : compare les descriptions pour trouver les services et compositions de services qui résolvent le problème.
- Descriptions possibles au niveau sémantique :
  - Nom des services, signatures, mot clés et/ou ontologies
  - Limites :
    - Nom : nécessité de connaître la nomenclature
      BLAS : SGEMM, E04-NAG : E04ABF
    - Signature : pas assez précis
      addition and multiplication : Matrix $\times$ Matrix $\rightarrow$ Matrix
    - Mot clés : ambiguë
      BLAS, SGEMM : $\alpha * A * B + \beta * C$
    - Ontologies :
      Pas de contrôle du solveur.

But

1. L'utilisateur exprime son problème aussi naturellement que possible.
2. L'algorithme trouve tous les services et compositions de services qui répondent au problème, en spécifiant les valeurs des paramètres.
3. La solution la plus avantageuse est choisie
   - interaction avec le middleware pour connaître les disponibilités
   - utilisation d'un modèle de coût
   - machine e-learning
4. La meilleure solution est exécutée.
5. Le résultat est renvoyé à l'utilisateur.

Transparent pour l'utilisateur

But

- Indépendance du domaine d'application.
- Description
  - Accessible pour un non spécialiste des technologie sous-jacentes
  - Décrivant le rôle des paramètres fonctionnels

Description : spécification algébrique

- Proche des notations mathématiques usuelles
- Domaine : order-sorted signature
  - S : ensemble fini de sortes
  - $\leq$: ordre partiel sur S (sous-typage)
  - Σ : ensemble de symboles (constantes et fonctions typées)
    - $c : s$, avec $s \in S$
    - $f : s_1 \times ... \times s_n \rightarrow s_{n+1}$, avec $s_1, ..., s_n, s_{n+1} \in S$ et $n \geq 1$
- Services et requête : termes sur la signature

Description : Exemple

- Domaine
  $S = \{Scalar, Matrix\}$
  $\Sigma = \{$
  $0, I : Matrix$
  $0, 1 : Scalar$
  $+ : Matrix \times Matrix \rightarrow Matrix$
  $+ : Scalar \times Scalar \rightarrow Scalar$
  $* : Matrix \times Matrix \rightarrow Matrix$
  $* : Scalar \times Matrix \rightarrow Matrix$
  $* : Scalar \times Scalar \rightarrow Scalar$
  $\}$
- Services
  - $Matrix\ x,\ y : serv_1(x, y) = x + y$
  - $Matrix\ x,\ y : serv_2(x, y) = x * y$
  - $Scalar\ \alpha, Matrix\ x : serv_3(\alpha, x) = \alpha * x$
  - $Scalar\ \alpha, Matrix\ x,\ y,\ z : serv_4(\alpha, x, y, z) = \alpha * x * y + z$
- Requête : $Matrix\ a,\ b,\ c : a * b + c$

Description : équations

- Parmi les solutions, il y aura :
  - $serv_1(serv_2(a, b), c) \quad = (a * b) + c$
- mais aussi :
  - $serv_1(c, serv_2(a, b)) \quad = c + (a * b)$
  - $serv_4(1, a, b, c) \quad = 1 * a * b + c$
  - $serv_1(serv_2(a, b), serv_2(I, c)) \quad = (a * b) + (I * c)$
  - ...

Plus d'informations sont nécessaires :

- $Matrix\ x, Matrix\ y : x + y = y + x$
- $Matrix\ x : 1 * x = x$
- $Matrix\ x : I * x = x$

$\Rightarrow$ Des équations sont ajoutées à la description du domaine applicatif

L'ensemble des solutions

- L'ensemble des solutions :

$$\{(s, \sigma_A)\ \}$$

- Propriétés :

$$\forall (s, \sigma_A) \in \text{l'ensemble des solutions }, \sigma_A(s) =_E r$$

- Matching équationel :

  Unification : $\sigma(u) = \sigma(v)$

  Unification équationelle : $\sigma(u) =_E \sigma(v)$

  Matching équationel : $\sigma(u) =_E v$

Système de type

Contraintes :

- Système de type adapté aux fonctions surchargées avec sous-typage
- Typage partiel :
  - Services typés
  - Requête typée
  - Équations typées ou non

$\Rightarrow$ Système de type de Castagna pour le $\lambda\&-$calculus

Mécanisme de comparaison

- Ré-écriture
  - Outils : C*i*ME, Maude, Elan
  - Nécessite un ordre bien fondé sur les équations et une forme normale
- Codé "à la main"
  - Gallier and Snyder, *Complete Sets of Transformations for General E-Unification*, 1989.
  - Réutilisation du travail théorique
  - Preuve sur papier de la complétude et de la correction

Algèbre linéaire - Description

- Sortes
  - Characters (Char)
  - Scalars (Int, NzInt, Real, NzReal)
  - Vectors (Vector)
  - Matrices (Matrix, InvMatrix, SymMatrix, ...)
    - InvMatrix $<$ Matrix
    - SymMatrix $<$ Matrix
    - ...
- Opérateurs
  - $-1, 1 : \mathit{NzInt}$, $0 : \mathit{Int}$
  - $-1.0, 1.0 : \mathit{NzReal}$, $0.0 : \mathit{Real}$
  - $I : \mathit{InvMatrix}$, $O : \mathit{Matrix}$
  - $+ : \mathit{Matrix} \times \mathit{Matrix} \rightarrow \mathit{Matrix}$, $\mathit{Int} \times \mathit{Int} \rightarrow \mathit{Int}$,...
  - $- : \mathit{Matrix} \times \mathit{Matrix} \rightarrow \mathit{Matrix}$, $\mathit{Int} \times \mathit{Int} \rightarrow \mathit{Int}$,...
  - $* : \mathit{Matrix} \times \mathit{Matrix} \rightarrow \mathit{Matrix}$, $\mathit{Int} \times \mathit{Int} \rightarrow \mathit{Int}$,...
  - $\hat{}\,T : \mathit{Matrix} \rightarrow \mathit{Matrix}$, $\mathit{SymMatrix} \rightarrow \mathit{SymMatrix}$,...
  - $\hat{}\,-1 : \mathit{InvMatrix} \rightarrow \mathit{InvMatrix}$, ...
  - ...

Algèbre linéaire - Équations

- Équations
  - éléments neutres :
    - (a + 0)=a
    - (a + 0.0)=a
    - (1 * a)=a
    - (I * a)=a
    - ...
  - éléments absorbants :
    - (0.0 * a)=0.0
    - (0 * a)=0
    - (O * a)=O
    - ...
  - Distribution / factorisation :
    - ((a * b) ^-1)=((b ^-1) * (a ^-1))
    - ((a ^T) ^-1)=((a ^-1) ^T)
    - ((a + b) * c)=((a * c) + (b * c))
    - (a * (b + c))=((a * b) + (a * c))
    - ...
  - Associativité :
    - (a * (b * c))=((a * b) * c)
    - (a + (b + c))=((a + b) + c)
  - Autre :
    - (I ^T)=I
    - *SymMatrix a* : (a ^T)=a

Algèbre linéaire - bibliothèques

- BLAS
  - $sgemm(transa : Char, transb : Char, m : Int, n : Int, k : Int, \alpha : Real, A : Matrix, lda : Int, B : Matrix, ldb : Int, \beta : Real, C : Matrix, ldc : Int)$
    $C \leftarrow ((\alpha * (op(transa, A) * op(transb, B))) + (\beta * C))$
  - $strsm(side : Char, uplo : Char, transA : Char, diag : Char, m : Int, n : Int, \alpha : Real, A : TriInvMatrix, lda : Int, B : Matrix, ldb : Int)$
    Solve a linear system with the A matrix triangular.
- LAPACK
  - $spotrf(uplo : Char, A : SymDefPosMatrix, info : Int)$
    Cholesky factorization.
  - $slaswp(n : Int, A : Matrix, lda : Int, k1 : Int, k2 : Int, ipiv : Vector, incx : Int)$
    Swap of the row of a matrix.
  - $sgetrf(m : Int, n : Int, A : InvMatrix, lda : Int, ipiv : Vector, info : Int)$
    LU factorization.

Algèbre linéaire - bibliothèques

- BLAS
  - $sgemm(transa : Char, transb : Char, m : Int, n : Int, k : Int, \alpha : Real, A : Matrix, lda : Int, B : Matrix, ldb : Int, \beta : Real, C : Matrix, ldc : Int)$
    $C \leftarrow ((\alpha * (op(transa, A) * op(transb, B))) + (\beta * C))$
  - $strsm(side : Char, uplo : Char, transA : Char, diag : Char, m : Int, n : Int, \alpha : Real, A : TriInvMatrix, lda : Int, B : Matrix, ldb : Int)$
    Solve a linear system with the A matrix triangular.
- LAPACK
  - $spotrf(uplo : Char, A : SymDefPosMatrix, info : Int)$
    Cholesky factorization.
  - $slaswp(n : Int, A : Matrix, lda : Int, k1 : Int, k2 : Int, ipiv : Vector, incx : Int)$
    Swap of the row of a matrix.
  - $sgetrf(m : Int, n : Int, A : InvMatrix, lda : Int, ipiv : Vector, info : Int)$
    LU factorization.

Algèbre linéaire - Exemple 1

- Requête : $A,\ B,\ C : Matrix \quad A * (B * C)$
- Une solution :

```
p1=O;
sgemm('n', 'n', ?, ?, ?, 1.0, B, ?, C, ?, 1.0, p1, ? );
# p1←B*C
p2=O;
sgemm('n', 'n', ?, ?, ?, 1.0, A, ?, p1, ?, 1.0, p2, ? );
# p2←A*p1
p2;
```

Algèbre linéaire - Exemple 2

- Requête : $Ax = B$
- Une solution :

```
p1=A;
p2=?;
sgetrf(?, ?, p1, ?, p2, ? );
# p2←Lu factorization of A (A= P*L*U)
p3=B;
slaswp(?, p3, ?, ?, ?, p2, ? );
# p3←swap of the rows of B
p4= p3;
strsm('l', 'l', 'n', 'n', ?, ?, 1.0, p1, ?, p4, ? );
# solves L*x=p3; p4←x;
p5= p4;
strsm('l', 'u', 'n', 'n', ?, ?, 1.0, p1, ?, p5, ? );
# solves U*x=p4; p5←x;
p5;
```

Optimisation - Description

- Sorts :
  - Fonctions (*funVector→Vector*, ...)
  - Contraintes (*constraint*)
  - Éléments manipulés par les fonctions (*Real*, *Matrix*, ...)
- Opérateurs :
  - *min*, *max*, ...
  - $<=$, &, ...
  - $*$, $+$, ...
- Équations :
  - Constraints manipulations
- Bibliothèques :
  - Matlab optimization toolbox
  - E04-NAG (Numerical Algorithms Group)

Optimisation - Exemple

- Requête : minimisation d'une fonction f dérivable sur $[a, b]$
- Solutions :

Cas général

```
p1=?;
E04ABF(f, ?, ?, a, b, ?, ?, p1, ?);
p1;
```

En utilisant les propriétés de dérivabilité

```
p1=?;
E04BBF(f, ?, ?, a, b, ?, ?, p1, ?, ? );
p1;
```

Interaction entre algèbre linéaire et optimisation

- Requête : $\min_x \frac{1}{2} x^T (Y * N * Y) x + f^T x,\ A_{eq} x \leq b_{eq},\ Ax = b$
- Une solution :

```
p1=O;
sgemm('n', 'n', ?, ?, ?, 1.0, Y, ?, N, ?, 1.0, p1, ? );
# p1←Y*N
p2=O;
sgemm('n', 'n', ?, ?, ?, 1.0, p1, ?, Y, ?, 1.0, p2, ? );
# p2←p1*Y
p3 = quadprog(p2, f, A,b, Aeq, beq );
# solve the quadratic problem
p3;
```

Choisir la meilleure solution

- Modèle de coût
- Interaction avec un midlleware
- Machine learning
- Clusterisation

Paramètres non fonctionnels

- Il y a souvent des paramètres non fonctionnels, par exemple des paramètres qui spécifient :
  - des propriétés sur les autres paramètres ;
  - quelle fonctionnalités est réalisée (switch).
- Par exemple : *strsm* du BLAS :
  - M est le nombre de lignes du paramètre B;
  - *UPLO* spécifie si le paramètre A est une matrice triangulaire supérieure ou inférieure;
  - *SIDE* spécifie si le problème à résoudre est $op(A) * X = \alpha * B$ où $X * op(A) = \alpha * B$.
- Le langage de description a été étendu pour prendre en compte ces paramètres.

Algèbre linéaire - Retour sur l'exemple 1

- Requête : $A,\ B,\ C : Matrix \quad A * (B * C)$
- Une solution :

```
p1=O;
sgemm('n', 'n', nbRow(p1), nbCol(p1), nbCol(B), 1.0, B, nbRow(B), C,
nbRow(C), 1.0, p1, nbRow(p1) );
# p1←B*C
p2=O;
sgemm('n', 'n', nbRow(p2), nbCol(p2), nbCol(A), 1.0, A, nbRow(A), p1,
nbRow(p1), 1.0, p2, nbRow(p2) );
# p2←A*p1
p2;
```

Conclusion

- Recherche de composition de service basé sur une description de la fonctionnalité des services
- Choix de la description :
  - Domaine : order sorted signature + équations
  - Services et requête : termes sur la signature
  - Typage : surcharge + sous-typage
- Mécanisme de découverte : unification équationnelle
- Limites :
  - Efficacité
  - Domaine d'application très spécifique.