NRC as a formal model for expressing bioinformatics workflows

A. Gambin [1], J. Hidders [2], N. Kwasnikowska [3], S. Lasota [1], J. Sroka [1], J. Tyszkiewicz [1], J. Van den Bussche [3]
[1] Warsaw University, [2] University of Antwerp, [3] Hasselt University

Context

- bioinformatics workflows
  - network of data centered processing steps
- processing steps involve
  - large amounts of complex data
    - sequence files, BLAST reports
    - XML data
  - a variety of tools
    - EMBOSS suite, BioPerl scripts
    - webservices, Mascot searches

Problems

- workflows execute as a mix of automated scripts and manual intervention
  - difficult to maintain
- results are stored in ad-hoc ways, e.g. files, Excel sheets
  - difficult to manage

Existing solutions

- workflow execution engines
  - Kepler [2], Taverna [3]
- not based on a formal data model, or too complicated and not data oriented

Our contribution

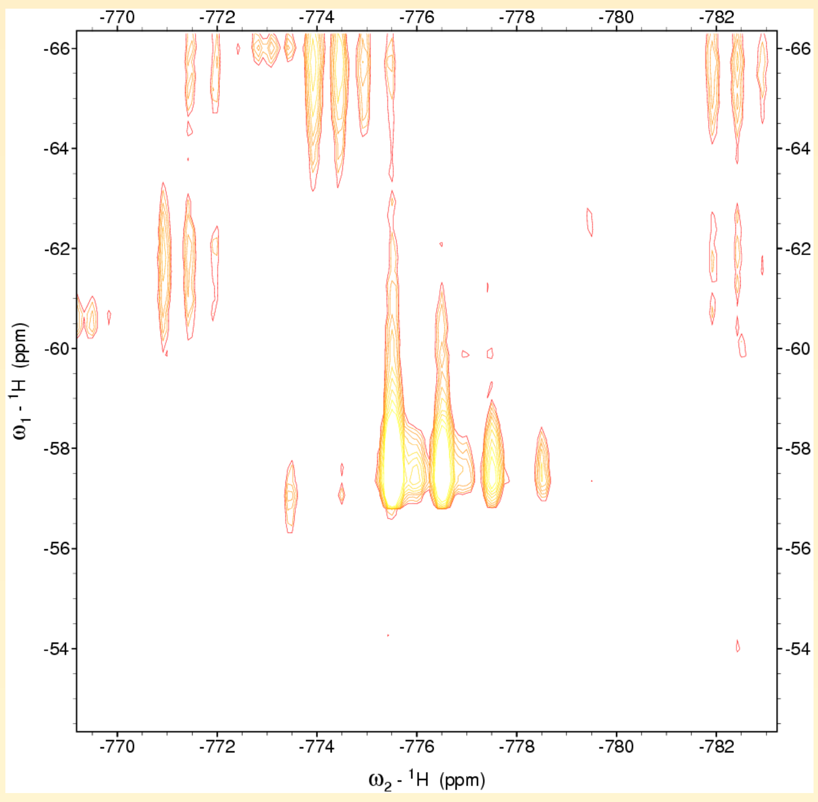
- using Nested Relational Calculus [1] for modeling data oriented workflows
- many bioinformatics workflows can be modeled in NRC
- advantages of using NRC
  - puts data oriented workflows on a firm foundation
  - formalism is already well understood
- natural approach
  - BioKleisli [4] is also based on NRC

Nested Relational Calculus

- established formalism for querying over complex objects [1]
- complex objects are arbitrarily nested collections and tuples
- collections can be sets, multi-sets and lists
- set-based model: sets {} and tuples ⟨⟩
- typed query language
  - extensible repertoire of base types
    - Boolean, String, Number
    - FASTA sequence file
    - XML, based on a DTD or XML Schema
  - complex types: nested sets and tuples

Workflow example – description

- 3D signal maps from LC-MS analysis of blood samples
- two groups: diseased and normal
- extracting clusters corresponding to peptides



- choosing classifiers, defined by a feature selection method
  - t-statistic, correlation
- and a classification algorithm
  - Decision Trees (DT), Random Forest (RF), Support Vector Machine (SVM)
- k-fold cross validation to obtain the following performance statistics for each classifier
  - sensitivity, specificity

Workflow example – data types

- base types

  String, Number, Boolean, Sample
- complex types
  - input type

    PatSample = ⟨ id: String, sample: Sample, diseased: Boolean ⟩
  - output type

    FSelCAlg = ⟨ tstat: {CAlg}, corr: {Calg} ⟩

    with CAlg = ⟨ dt: PStats, rf: PStats, svm: PStats ⟩

    and PStats = ⟨ sensitivity: Number, specificity: Number ⟩
  - auxiliary types

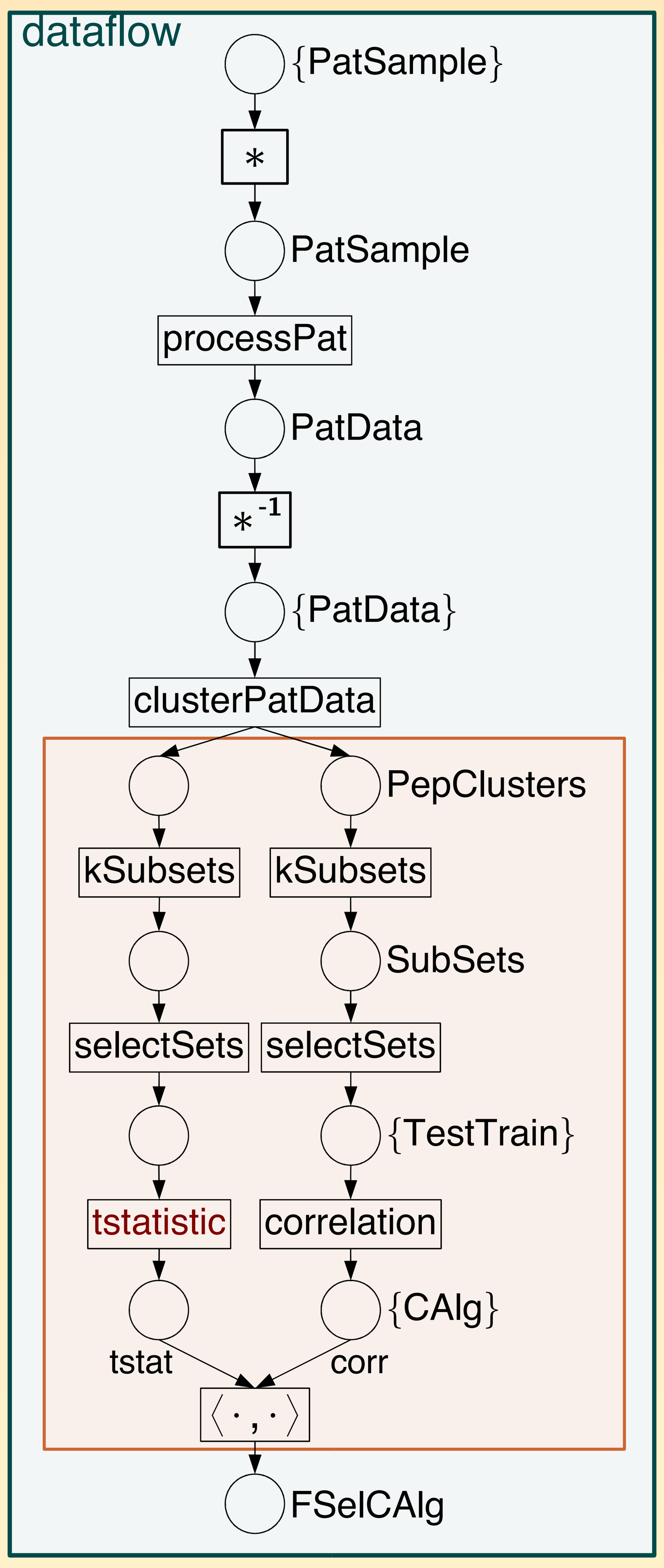
    TestTrain = ⟨ test: PepClusters, train: PepClusters ⟩

    PepClusters = { ⟨ clustermass: Number, patlist: PatList ⟩ }

    PatList = { ⟨ patid: String, diseased: Boolean, intensity: Number ⟩ }

NRC core operations

- constant value of a base type — "John", true, 89
- variable of any type, either base type or complex — $patient
- tuple construction — ⟨ name: "John", condition: true, age: 89 ⟩
- tuple projection — $patient.name
- empty set construction — ∅
- singleton set construction — {$patient}
- set union — $patientList = $healthy ∪ $diseased
- flattening of a nested set
  - $patientList = flatten({$healthy, $diseased})
- iteration over a set
  - for $patient in $patientList return $patient.name
- named program definition — pBLAST: FASTA → {AccessionNr}
  - external programs, used as a "black box"
  - internal programs, help with top-down design
- equality test for base types — $patient.name = "John"
- emptiness test for sets — $patientList = ∅
- conditional
  - if $p.condition then $diseased ∪ {$p} else $healthy ∪ {$p}
- core operations can be combined into programs



Workflow example – NRC programs

- top-down design of the workflow
  - after processing and clustering of raw patient data, k-fold cross validation is performed

```
define dataflow($s: {PatSample}): FSelCAlg as
 kfoldCrossValidation(
  clusterPatData(
   for $i in $s return processPat($i)))
```

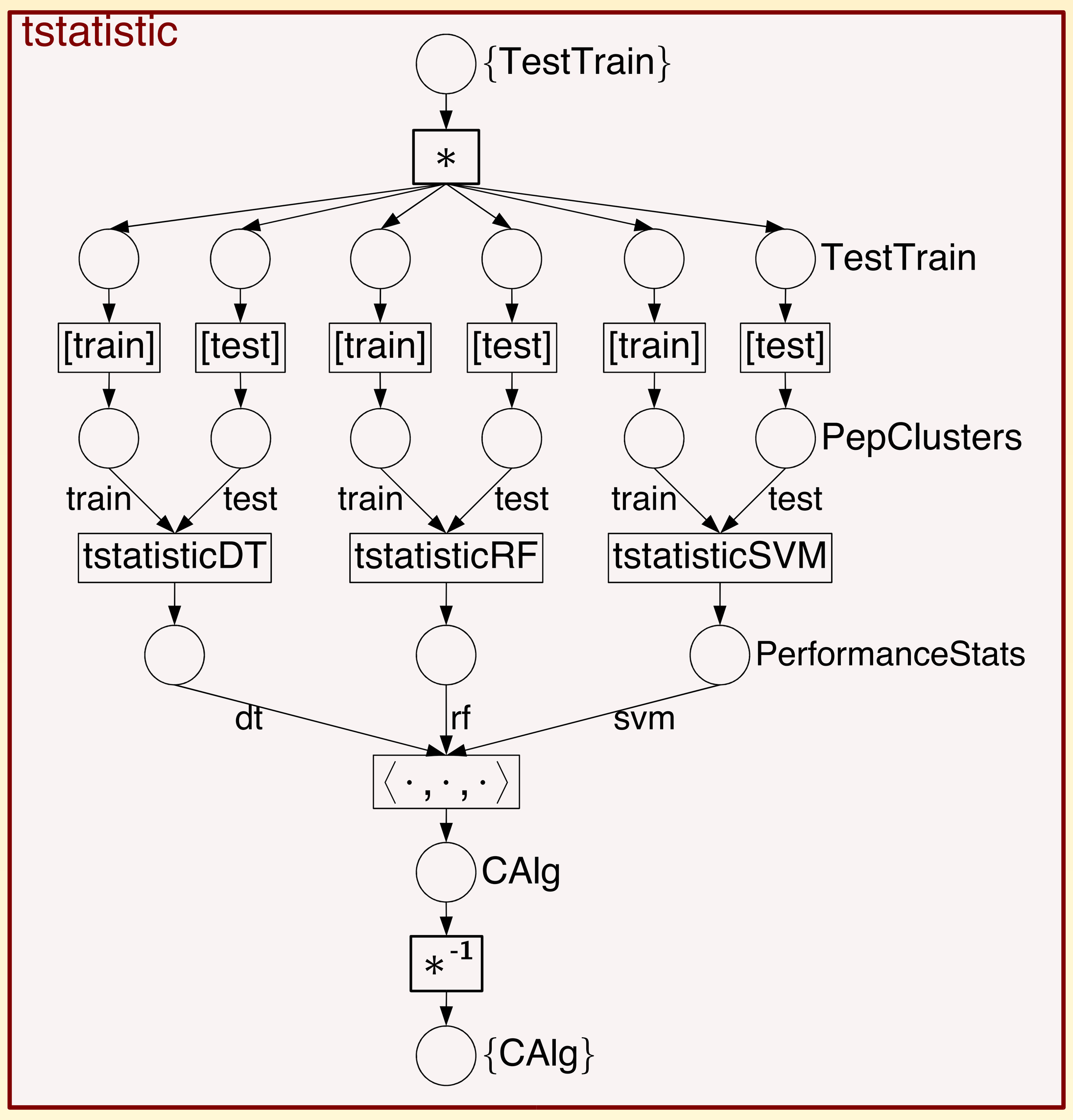
- some internal programs
  - the set of peptide clusters is divided k times into a training set and a test set by internal program selectSets and external program kSubsets

    these pairs are passed to internal programs tstatistic and correlation, then a tuple is constructed from their results

```
define kfoldCrossValidation($pc: PepClusters): FSelCAlg as
⟨ tstat: tstatictic(selectSets(kSubsets($pc))),
  corr: correlation(selectSets(kSubsets($pc))) ⟩
```

  - for each pair of training set and test set, external programs corresponding to chosen classifiers with t-statistic as feature selection method are invoked, then a tuple is constructed from received results

```
define tstatistic($tt: {TestTrain}): {Calg} as
 for $i in $tt return
 ⟨ dt: tstatisticDT($i.train, $i.test),
   rf: tstatisticRF($i.train, $i.test),
   svm: tstatisticSVM($i.train, $i.test) ⟩
```



Graphical representation

- dataflow language based on Petri nets and Nested Relational Calculus [6]
- typing system and core operations from NRC
- top-down, hierarchical design of the data flow

Acknowledgements

Special thanks to J. Dutkowski, A. Gambin, B. Kluge, K. Kowalczyk, J.Tiuryn from Institute of Informatics, Warsaw University, and M. Dadlez from Institute of Biochemistry and Biophysics, Polish Academy of Science, for providing their workflow.

References

[1] P. Buneman et al., "Principles of programming with complex objects and collection types", Theoretical Computer Science, 1995, 149:3–48.

[2] I. Attinas et al., "Kepler: An Extensible System for Design and Execution of Scientific Workflows", in proc. SSDBM 2004.

[3] T. Oinn et al., "Taverna: a tool for the composition and enactment of bioinformatics workflows", Bioinformatics, 2004, 20(17):3045–3054.

[4] S. Davidson et al., "BioKleisli: A Digital Library for Biomedical Researchers", J. Digital Libraries, 1997, 1(1):36–53.

[5] A. Tröger et al., "A language for comprehensively supporting the in vitro experimental process in silico", in proc. BIBE 2004.

[6] J. Hidders et al., "Petri nets + nested relational calculus = dataflow language", submitted to CoopIS 2005.

NRC as a formal model for expressing bioinformatics workflows

A. Gambin [1], J. Hidders [2], N. Kwasnikowska [3], S. Lasota [1], J. Sroka [1], J. Tyszkiewicz [1], J. Van den Bussche [3]
[1] Warsaw University, [2] University of Antwerp, [3] Hasselt University

Context

- bioinformatics workflows
  - network of data centered processing steps
- processing steps involve
  - large amounts of complex data
    - sequence files, BLAST reports
    - XML data
  - a variety of tools
    - EMBOSS suite, BioPerl scripts
    - webservices, Mascot searches

Problems

- workflows execute as a mix of automated scripts and manual intervention
  - difficult to maintain
- results are stored in ad-hoc ways, e.g. files, Excel sheets
  - difficult to manage

Existing solutions

- workflow execution engines
  - Kepler [2], Taverna [3]
- not based on a formal data model, or too complicated and not data oriented

Our contribution

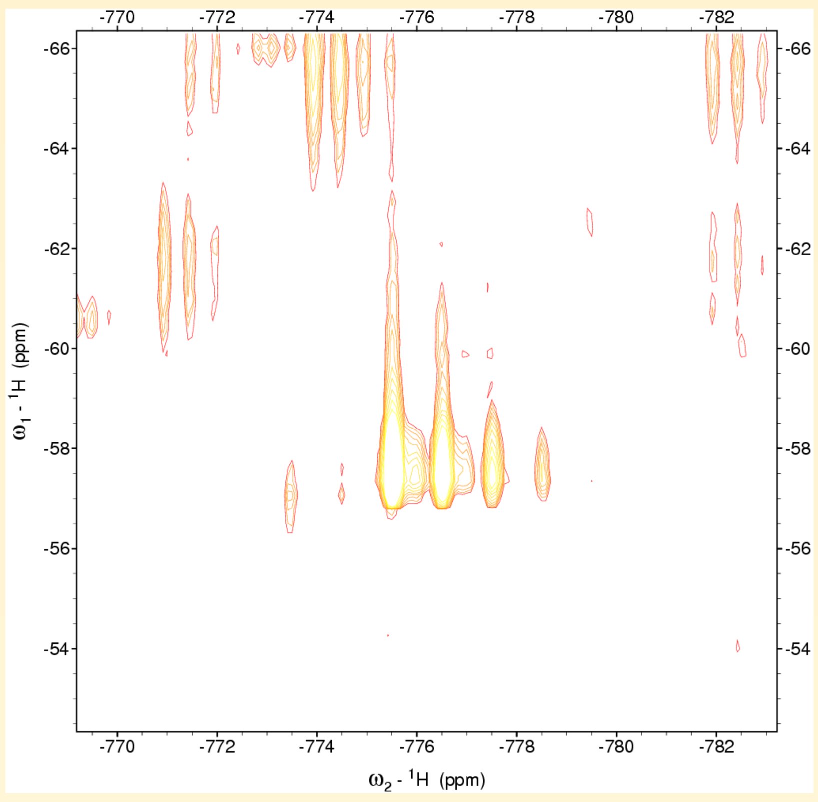
- using Nested Relational Calculus [1] for modeling data oriented workflows
- many bioinformatics workflows can be modeled in NRC
- advantages of using NRC
  - puts data oriented workflows on a firm foundation
  - formalism is already well understood
- natural approach
  - BioKleisli [4] is also based on NRC

Nested Relational Calculus

- established formalism for querying over complex objects [1]
- complex objects are arbitrarily nested collections and tuples
- collections can be sets, multi-sets and lists
- set-based model: sets {} and tuples ⟨⟩
- typed query language
  - extensible repertoire of base types
    - Boolean, String, Number
    - FASTA sequence file
    - XML, based on a DTD or XML Schema
  - complex types: nested sets and tuples

Workflow example – description

- 3D signal maps from LC-MS analysis of blood samples
- two groups: diseased and normal
- extracting clusters corresponding to peptides



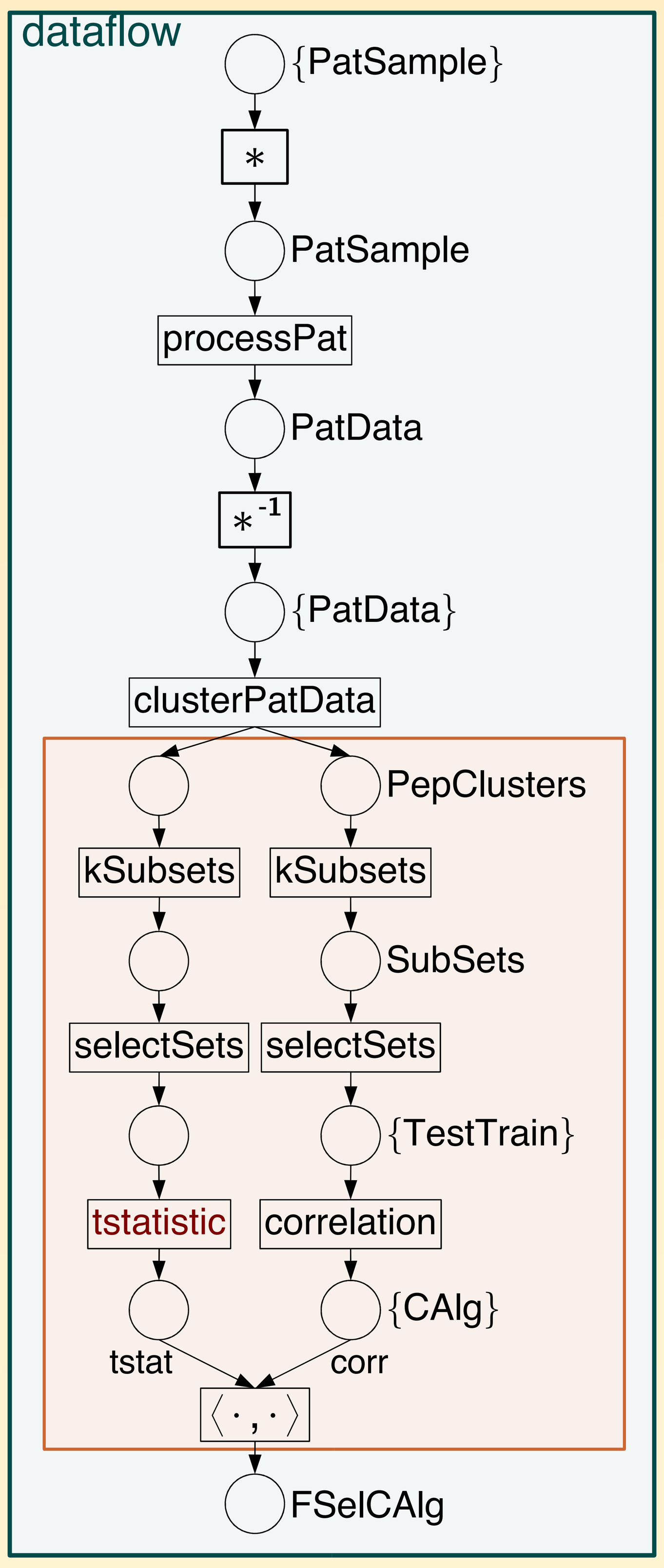
- choosing classifiers, defined by a feature selection method
  - t-statistic, correlation
- and a classification algorithm
  - Decision Trees (DT), Random Forest (RF), Support Vector Machine (SVM)
- k-fold cross validation to obtain the following performance statistics for each classifier
  - sensitivity, specificity

Workflow example – data types

- base types
  String, Number, Boolean, Sample
- complex types
  - input type
    PatSample = ⟨ id: String, sample: Sample, diseased: Boolean ⟩
  - output type
    FSelCAlg = ⟨ tstat: {CAlg}, corr: {Calg} ⟩
    with CAlg = ⟨ dt: PStats, rf: PStats, svm: PStats ⟩
    and PStats = ⟨ sensitivity: Number, specificity: Number ⟩
  - auxiliary types
    TestTrain = ⟨ test: PepClusters, train: PepClusters ⟩
    PepClusters = { ⟨ clustermass: Number, patlist: PatList ⟩ }
    PatList = { ⟨ patid: String, diseased: Boolean, intensity: Number ⟩ }

NRC core operations

- constant value of a base type — "John", true, 89
- variable of any type, either base type or complex — $patient
- tuple construction — ⟨ name: "John", condition: true, age: 89 ⟩
- tuple projection — $patient.name
- empty set construction — ∅
- singleton set construction — {$patient}
- set union — $patientList = $healthy ∪ $diseased
- flattening of a nested set
  - $patientList = flatten({$healthy, $diseased})
- iteration over a set
  - for $patient in $patientList return $patient.name
- named program definition — pBLAST: FASTA → {AccessionNr}
  - external programs, used as a "black box"
  - internal programs, help with top-down design
- equality test for base types — $patient.name = "John"
- emptiness test for sets — $patientList = ∅
- conditional
  - if $p.condition then $diseased ∪ {$p} else $healthy ∪ {$p}
- core operations can be combined into programs



Workflow example – NRC programs

- top-down design of the workflow
  - after processing and clustering of raw patient data, k-fold cross validation is performed

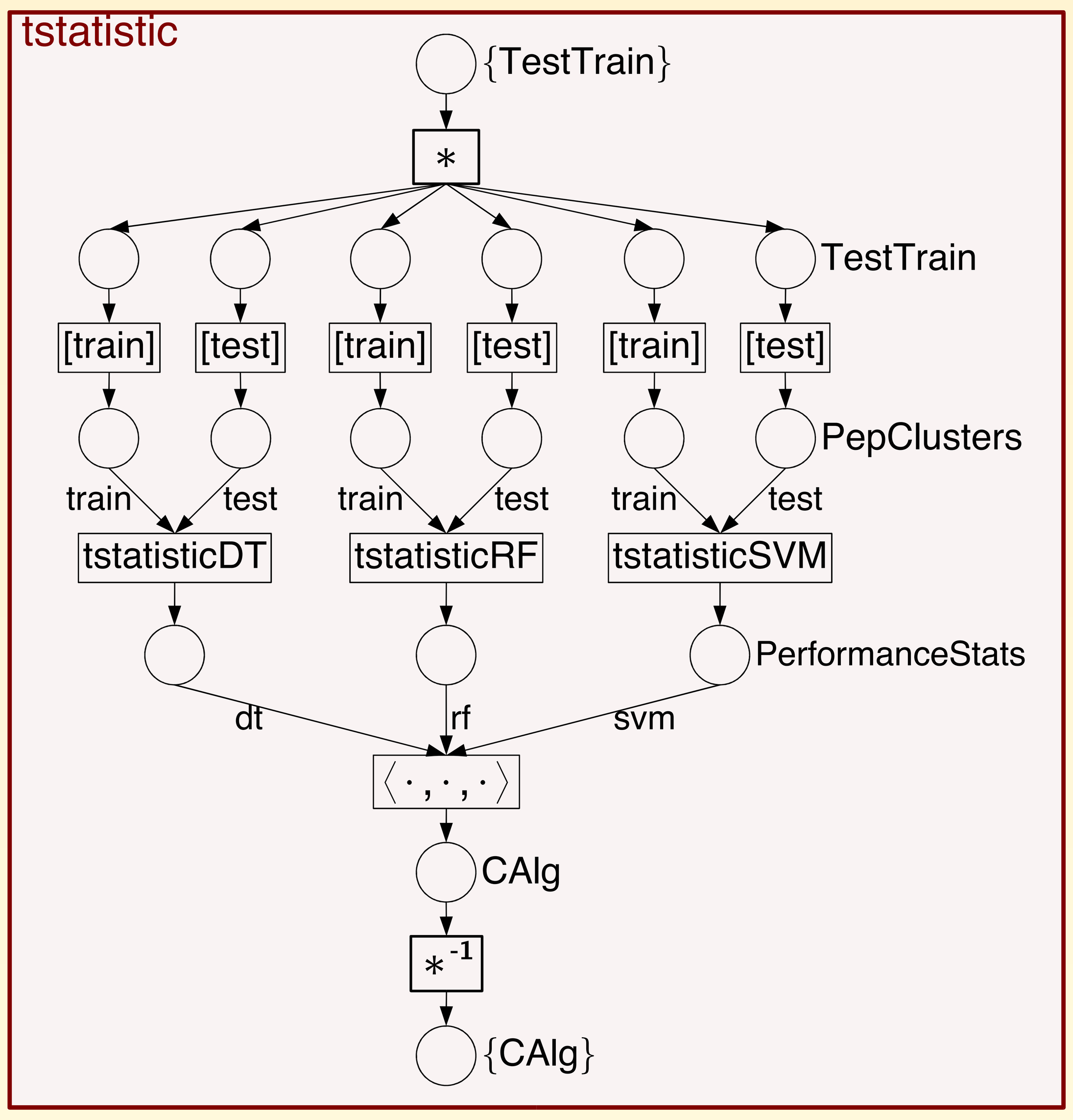
```
define dataflow($s: {PatSample}): FSelCAlg as
  kfoldCrossValidation(
    clusterPatData(
      for $i in $s return processPat($i)))
```

- some internal programs
  - the set of peptide clusters is divided k times into a training set and a test set by internal program selectSets and external program kSubsets
    these pairs are passed to internal programs tstatistic and correlation, then a tuple is constructed from their results

```
define kfoldCrossValidation($pc: PepClusters): FSelCAlg as
 ⟨ tstat: tstatictic(selectSets(kSubsets($pc))),
   corr: correlation(selectSets(kSubsets($pc))) ⟩
```

  - for each pair of training set and test set, external programs corresponding to chosen classifiers with t-statistic as feature selection method are invoked, then a tuple is constructed from received results

```
define tstatistic($tt: {TestTrain}): {Calg} as
  for $i in $tt return
   ⟨ dt: tstatisticDT($i.train, $i.test),
     rf: tstatisticRF($i.train, $i.test),
     svm: tstatisticSVM($i.train, $i.test) ⟩
```



Graphical representation

- dataflow language based on Petri nets and Nested Relational Calculus [6]
- typing system and core operations from NRC
- top-down, hierarchical design of the data flow

Acknowledgements

Special thanks to J. Dutkowski, A. Gambin, B. Kluge, K. Kowalczyk, J.Tiuryn from Institute of Informatics, Warsaw University, and M. Dadlez from Institute of Biochemistry and Biophysics, Polish Academy of Science, for providing their workflow.

References

[1] P. Buneman et al., "Principles of programming with complex objects and collection types", Theoretical Computer Science, 1995, 149:3–48.

[2] I. Attinas et al., "Kepler: An Extensible System for Design and Execution of Scientific Workflows", in proc. SSDBM 2004.

[3] T. Oinn et al., "Taverna: a tool for the composition and enactment of bioinformatics workflows", Bioinformatics, 2004, 20(17):3045–3054.

[4] S. Davidson et al., "BioKleisli: A Digital Library for Biomedical Researchers", J. Digital Libraries, 1997, 1(1):36–53.

[5] A. Tröger et al., "A language for comprehensively supporting the in vitro experimental process in silico", in proc. BIBE 2004.

[6] J. Hidders et al., "Petri nets + nested relational calculus = dataflow language", submitted to CoopIS 2005.

NRC as a formal model for expressing bioinformatics workflows

A. Gambin [1], J. Hidders [2], N. Kwasnikowska [3], S. Lasota [1], J. Sroka [1], J. Tyszkiewicz [1], J. Van den Bussche [3]
[1] Warsaw University, [2] University of Antwerp, [3] Hasselt University

Context

- bioinformatics workflows
  - network of data centered processing steps
- processing steps involve
  - large amounts of complex data
    - sequence files, BLAST reports
    - XML data
  - a variety of tools
    - EMBOSS suite, BioPerl scripts
    - webservices, Mascot searches

Problems

- workflows execute as a mix of automated scripts and manual intervention
  - difficult to maintain
- results are stored in ad-hoc ways, e.g. files, Excel sheets
  - difficult to manage

Existing solutions

- workflow execution engines
  - Kepler [2], Taverna [3]
- not based on a formal data model, or too complicated and not data oriented

Our contribution

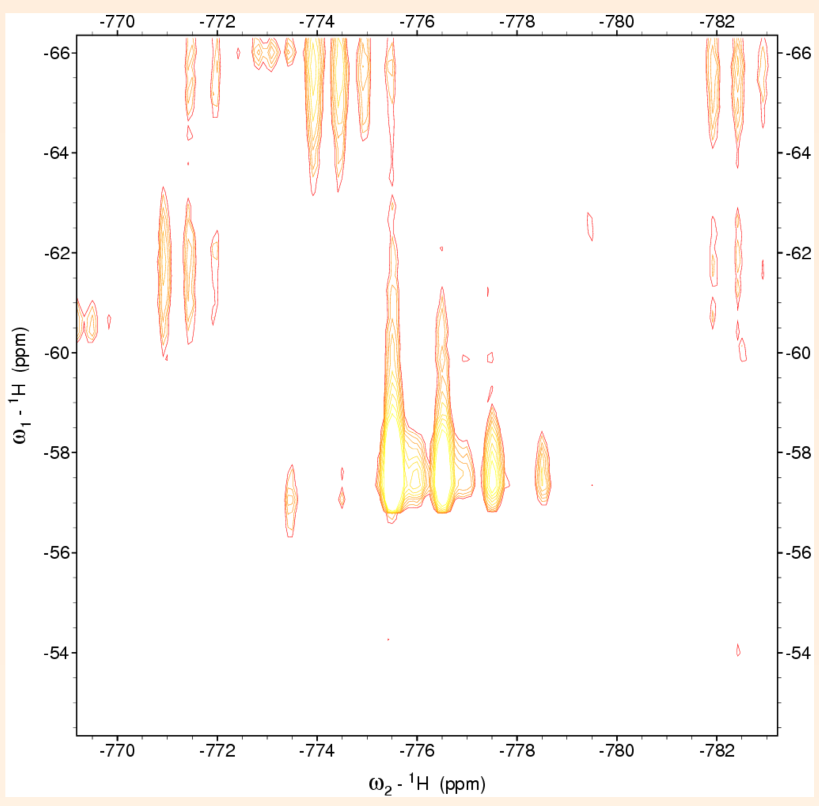
- using Nested Relational Calculus [1] for modeling data oriented workflows
- many bioinformatics workflows can be modeled in NRC
- advantages of using NRC
  - puts data oriented workflows on a firm foundation
  - formalism is already well understood
- natural approach
  - BioKleisli [4] is also based on NRC

Nested Relational Calculus

- established formalism for querying over complex objects [1]
- complex objects are arbitrarily nested collections and tuples
- collections can be sets, multi-sets and lists
- set-based model: sets {} and tuples ⟨⟩
- typed query language
  - extensible repertoire of base types
    - Boolean, String, Number
    - FASTA sequence file
    - XML, based on a DTD or XML Schema
  - complex types: nested sets and tuples

Workflow example – description

- 3D signal maps from LC-MS analysis of blood samples
- two groups: diseased and normal
- extracting clusters corresponding to peptides



- choosing classifiers, defined by a feature selection method
  - t-statistic, correlation
- and a classification algorithm
  - Decision Trees (DT), Random Forest (RF), Support Vector Machine (SVM)
- k-fold cross validation to obtain the following performance statistics for each classifier
  - sensitivity, specificity

Workflow example – data types

- base types

  String, Number, Boolean, Sample
- complex types
  - input type

    PatSample = ⟨ id: String, sample: Sample, diseased: Boolean ⟩
  - output type

    FSelCAlg = ⟨ tstat: {CAlg}, corr: {CAlg} ⟩

    with CAlg = ⟨ dt: PStats, rf: PStats, svm: PStats ⟩

    and PStats = ⟨ sensitivity: Number, specificity: Number ⟩
  - auxiliary types

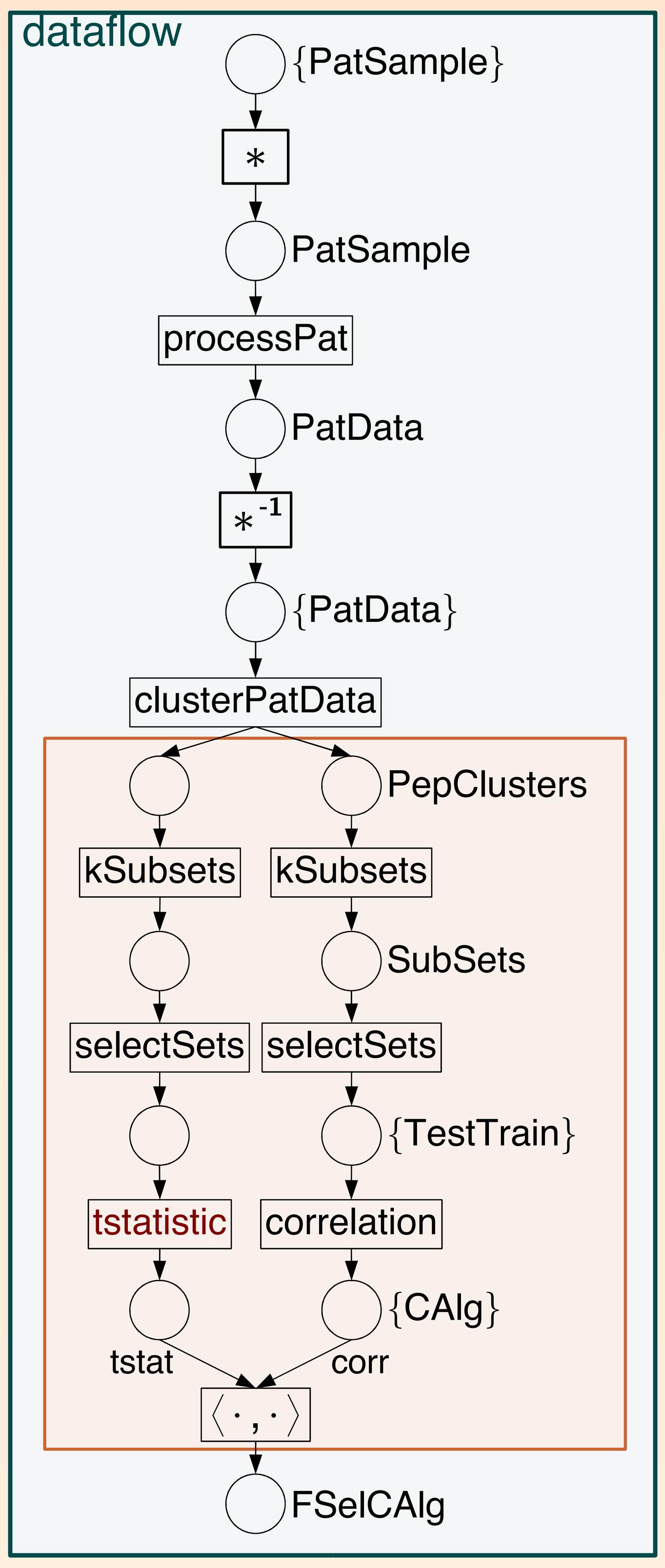
    TestTrain = ⟨ test: PepClusters, train: PepClusters ⟩

    PepClusters = { ⟨ clustermass: Number, patlist: PatList ⟩ }

    PatList = { ⟨ patid: String, diseased: Boolean, intensity: Number ⟩ }

NRC core operations

- constant value of a base type — "John", true, 89
- variable of any type, either base type or complex — $patient
- tuple construction — ⟨ name: "John", condition: true, age: 89 ⟩
- tuple projection — $patient.name
- empty set construction — ∅
- singleton set construction — {$patient}
- set union — $patientList = $healthy ∪ $diseased
- flattening of a nested set
  - $patientList = flatten({$healthy, $diseased})
- iteration over a set
  - for $patient in $patientList return $patient.name
- named program definition — pBLAST: FASTA → {AccessionNr}
  - external programs, used as a "black box"
  - internal programs, help with top-down design
- equality test for base types — $patient.name = "John"
- emptiness test for sets — $patientList = ∅
- conditional
  - if $p.condition then $diseased ∪ {$p} else $healthy ∪ {$p}
- core operations can be combined into programs



Workflow example – NRC programs

- top-down design of the workflow
  - after processing and clustering of raw patient data, k-fold cross validation is performed

```
define dataflow($s: {PatSample}): FSelCAlg as
  kfoldCrossValidation(
    clusterPatData(
      for $i in $s return processPat($i)))
```

- some internal programs
  - the set of peptide clusters is divided k times into a training set and a test set by internal program selectSets and external program kSubsets

    these pairs are passed to internal programs tstatistic and correlation, then a tuple is constructed from their results

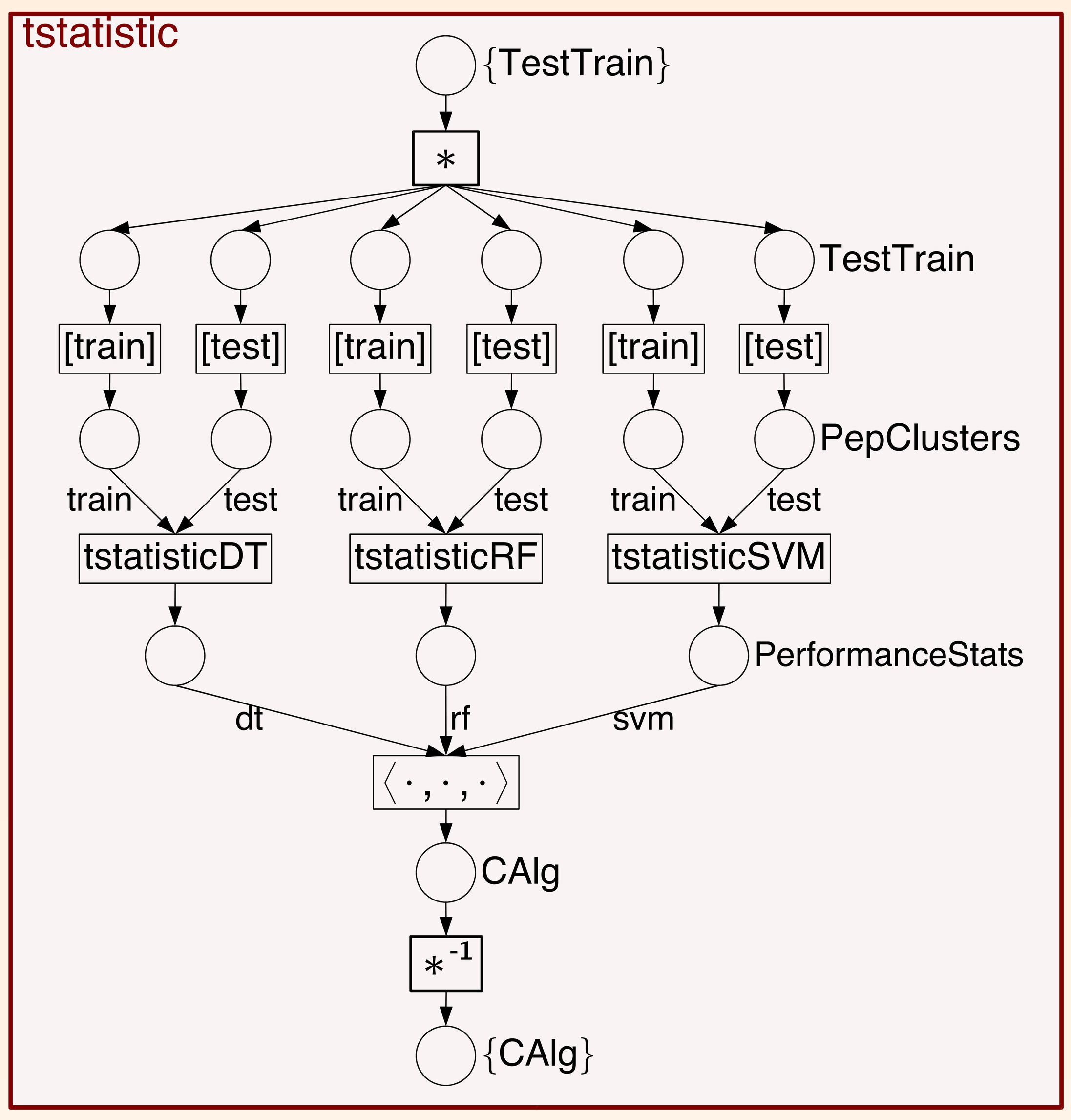
```
define kfoldCrossValidation($pc: PepClusters): FSelCAlg as
 ⟨ tstat: tstatictic(selectSets(kSubsets($pc))),
   corr: correlation(selectSets(kSubsets($pc))) ⟩
```

  - for each pair of training set and test set, external programs corresponding to chosen classifiers with t-statistic as feature selection method are invoked, then a tuple is constructed from received results

```
define tstatistic($tt: {TestTrain}): {Calg} as
 for $i in $tt return
  ⟨ dt: tstatisticDT($i.train, $i.test),
    rf: tstatisticRF($i.train, $i.test),
    svm: tstatisticSVM($i.train, $i.test) ⟩
```

Graphical representation

- dataflow language based on Petri nets and Nested Relational Calculus [6]
- typing system and core operations from NRC
- top-down, hierarchical design of the data flow



Acknowledgements

Special thanks to J. Dutkowski, A. Gambin, B. Kluge, K. Kowalczyk, J.Tiuryn from Institute of Informatics, Warsaw University, and M. Dadlez from Institute of Biochemistry and Biophysics, Polish Academy of Science, for providing their workflow.

References

[1] P. Buneman et al., "Principles of programming with complex objects and collection types", Theoretical Computer Science, 1995, 149:3–48.

[2] I. Attinas et al., "Kepler: An Extensible System for Design and Execution of Scientific Workflows", in proc. SSDBM 2004.

[3] T. Oinn et al., "Taverna: a tool for the composition and enactment of bioinformatics workflows", Bioinformatics, 2004, 20(17):3045–3054.

[4] S. Davidson et al., "BioKleisli: A Digital Library for Biomedical Researchers", J. Digital Libraries, 1997, 1(1):36–53.

[5] A. Tröger et al., "A language for comprehensively supporting the in vitro experimental process in silico", in proc. BIBE 2004.

[6] J. Hidders et al., "Petri nets + nested relational calculus = dataflow language", submitted to CoopIS 2005.