Query Combinators for Medical Informatics

Clark C. Evans, Kyrylo Siminov, PhD
Prometheus Research, LLC, New Haven, CT

Introduction

The discovery and management of new knowledge relating to health and disease is quite often hindered by technical complexities of querying and transforming data. Even though SQL has evolved over the past 44 years to respond to new challenges, it is poorly suited to querying clinical data. There have been two notable improvements in this space. LINQ has introduced the notion of monadic containers; although unlike SQL, LINQ must be used within the context of a full-blown programming language. Xpath has introduced the notion of navigation via combinators, a way to explore data that users find intuitive; unfortunately, its combinator algebra is limited to hierarchical data and cannot handle aggregation and general transformations. This Query Combinator approach is an extensible query language, formally defined with a combinator algebra, that addresses these deficiencies.

Learning By Example

This poster demonstrates a taste of this approach by showing how elementary queries and combinators form an algebra to make rather simple looking, albeit sophisticated composite queries.

Elementary Elements

Query	*Signature*	*Description*
`patient`	Database → Patient*	List all individuals in the database.
`observation`	Patient → Observation*	For a patient, list all observations.

Algebraic Operators

Combinator	*Description*
`count(A→B*) ⇒ (A→Int)`	For any query, produce a query with the same input domain, having an integer output domain. The value of that integer is the count of the correlated records.
`(A→B*).(B→C*) ⇒(A→C*)`	For any two queries with signatures `(A→B*)` and `(B→C*)`, produce a query with signature query `(A→C*)`, navigating from `A` to `C` though correlated `B`'s.

Composed Queries

Query	*Signature*	*Description*
`count(patient)`	Database → Integer	Count all individuals in the database.
`patient.observation`	Database → Observation*	List observations in the database, by patient.
`count(patient.observation)`	Database → Integer*	Count number of observations in the database.

Conclusion

While the example queries above are admittedly unimpressive, the approach has been used to produce viable answers to significant medical research questions on data from electronic health records databases, such as exploring cases where the intensification of hypertension medication was followed by a drop in blood pressure. More details can be found in the paper listed below. An MIT/Apache licensed version of implementation of Query Combinators for the Julia Language is available on github at https://github.com/rbt-lang/DataKnots.jl

References

1. Evans CE, Simonov K, Query Combinators (https://arxiv.org/abs/1702.08409)