Introduction to Research in Data Science

October 23, 2014
James Cheney
Big Data
No Big Data
Wicked Data

**wicked problem**, n. a problem that is difficult or impossible to solve because of incomplete, contradictory, and changing requirements that are often difficult to recognize
Research questions:
Language-integrated query

How can we (safely/securely) program multiple layers (database, browser, regular PL)?
A LINQ example

```csharp
let rec canDoAll(tsks) =
  match tsks with
  [] -> @ fun name -> true @
| tsk::tsks' -> @ fun name ->
  (%canDo) name tsk && (%canDoAll tsks') name @

query {
  for x in employees
  where ((%canDoAll ["build","call"] x.name)
  yield {name=x.name} }
```
LINQ example

```
let rec canDoAll(tsks) =
  match tsks with
  | [] -> <> fun name -> true <>
  | tsk::tsks' -> <> fun name ->
    (canDo name tsk && (canDoAll tsks') name <>)
query {
  for x in employees
  where ((canDoAll ["build","call"] x.name) yield {name=x.name} }
```
Example

let elem = <@ fun x xs ->
    query { for y in xs
             exists (y=x) } @>

let canDo = <@ fun name tsk ->
    (%elem) tsk (for t in tasks
             where (t.emp = name)
             yield t.tsk ) @>

query { for x in employees
      where ( (%canDo) x.name "build")
      yield {name = x.name}
Example

```ocaml
let elem = <@ fun x xs -> query { for y in xs exists (y=x) } @>

let canDo = <@ fun name tsk ->
  (fun x xs ->
    query { for y in xs exists (y=x) }
    tsk (for t in tasks where (t.emp = name)
      yield t.tsk ) @>

query { for x in employees
  where ((%canDo) x.name "build")
  yield {name = x.name}
```
Example

```ocaml
let elem = <@ fun x xs ->
  query { for y in xs
            exists (y=x) } @>

let canDo = <@ fun name tsk ->
  (fun x xs ->
    query { for y in xs
             exists (y=x) }
    tsk (for t in tasks
         where (t.emp = name)
         yield t.tsk )
  ) x.name "build"
  yield {name = x.name} @>

query { for x in employees
         where ((%canDo) x.name "build")
         yield {name = x.name} }
Example

Let's consider the following code snippet:

```ml
let elem = <@ fun x xs ->
  query { for y in xs
          exists (y=x) } @>

let canDo = <@ fun name tsk ->
  query { for x in employees
          where ((%elem) tsk (for t in tasks
                         where (t.emp = name)
                               yield t.tsk ))
          yield {name = x.name} @>
```

This is what LINQ normally sees.

X (failure or query avalanche)
Example

```ml
let elem = <@ fun x xs ->
    query { for y in xs
    exists (y=x) } @>

let canDo = <@ fun name tsk ->
    query { for t in tasks
    where (t.emp = name)
    yield t.tsk } @>

query { for x in employees
    where ((%canDo) x.name "build")
    yield {name = x.name} 
```
Example

let elem = <@ fun x xs ->
    query { for y in xs
        exists (y=x) }
@>

let canDo = <@ fun name tsk ->
    (fun x xs ->
        query { for y in xs
            exists (y=x) }
    ) tsk
        (for t in tasks
            where (t.emp = name)
            yield t.tsk )
@>

query { for x in employees
    where ((fun name tsk ->
        query { for y in (for t in tasks
            where (t.emp = name)
            yield t.tsk )
        exists (y= tsk) }
    ) x.name "build")
    yield {name = x.name}
Example

```plaintext
let elem = <@ fun x xs ->
    query { for y in xs
        exists (y=x) }
@>

let canDo = <@ fun name tsk ->
    (fun x xs ->
    query { for y in xs
        exists (y=x) }
    tsk (for t in tasks
        where (t.emp = name)
        yield t.tsk )
@>

query { for x in employees
    where ((fun name tsk ->
        (fun x xs ->
        query { for y in xs
            exists (y=x) }
        tsk (for t in tasks
            where (t.emp = name)
            yield t.tsk )
        exists (y= "build") )
    )
    yield {name = x.name}
```
Example

```ml
let elem = <@ fun x xs ->
  query { for y in xs
    exists (y=x) } @>

let canDo = <@ fun name tsk ->
  (elem) tsk (for t in tasks
    where (t.emp = name)
    yield t.tsk ) @>

query { for x in employees
  where ((canDo) x.name "build")
  yield {name = x.name} }
```
SELECT x.name
FROM employees x
WHERE EXISTS (SELECT t.tsk FROM tasks t WHERE t.emp = x.name)
Research questions: Data transformation

- How do I make use of data in format X with tools that expect Y?
- What if some of X is missing or Y requires information that X doesn't provide?
Bidirectional transformations

• Can synchronize two data sources using functions:
  • $get : A \rightarrow B$, $put : A \rightarrow B \rightarrow A$

• satisfying laws: $put(get a) = a$, $get(put a b) = b$

• generalizing view updates in databases

• Current projects:
  • bidirectional transformations with effects
    • extending classical framework to allow monadic effects
    • (e.g. $put : A \rightarrow B \rightarrow MA$)

• investigating "least change" principles
  • give change to one side, minimize the "damage" done to the other side
Research questions: Provenance

How can we trust the results of large programs (P(bugs) = 1.0) running on large infrastructure where failure during run is expected over large amounts of uncertain/noisy data?

A Scientist’s Nightmare: Software Problem Leads to Five Retractions

Until recently, Geoffrey Chang’s career was on a trajectory most young scientists only dream about. In 1999, at the age of 28, the protein crystallographer landed a faculty position at the prestigious Scripps Research Institute in San Diego, California. The next year, in a ceremony at the White House, Chang received a 2001 Science paper, which described the structure of a protein called MsbA, isolated from the bacterium Escherichia coli. MsbA belongs to a huge and ancient family of molecules that use energy from adenosine triphosphate to transport molecules across cell membranes. These so-called ABC transporters perform many
Foundations for trust and accountability

• Provenance and annotation
  • Understanding derivation process / history of data
• Seems to mean different things in different settings:
  • to DB people: semiring interpretations of relational algebra
  • to PL/security people: information flow, dependency tracking
  • to scientists: version management / smart replay / quality control for data (and derived results)
• Are we solving the right problem(s)?
  • Many ad hoc solutions; few specifications or "correct" implementations
• My view: mathematical foundations/attempts to specify problems essential to progress
Mathematical foundations of provenance

- Dependency: understanding how outputs depend on inputs (see also: noninterference in security)
  - "if I change this part, what parts of output will/may change?"
- Explanation: Galois connections between "parts" of input and output
  - "what part of input was needed to force this part of the output to be 42?"
- View maintenance/incremental computation
  - "if I change this part, how can I recompute the output most efficiently?"
- Bidirectionality/view update
  - "if I want to change this part of the output, what input changes could do this?"
- justifications/witnesses (why-provenance)
- reasoning about knowledge/uncertainty (multi-modal logic)
- mathematical modeling of causality (see also: Bayes nets)
Summary

• My work explores the interaction between language design, semantics, and data management.

• Emphasis on applying formal foundations
  • particularly concepts from programming language semantics

• to improve understanding of, address "wicked problems" coming from scientists working with data
  • Programmability, data versioning/synchronization, provenance/accountability
  • Relevant to "science data", though maybe not what people currently think of as mainstream "data science"