Violating dataflow and getting away with it

Mistral Contrastin Andrew Rice

Mistral.Contrastin@cl.cam.ac.uk
dodisturb.me
@madgen_

UNIVERSITY OF CAMBRIDGE
QUEENS' COLLEGE CAMBRIDGE

Datalog syntax in this talk

```
advisor("Euler", "Lagrange").
advisor("Lagrange", "Fourier").
advisor("Lagrange", "Poisson").
```

Fact

Predicate **Atom (atomic formula)**

```
academicAncestor(X,Y) :- advisor(X,Y).
academicAncestor(X,Y) :- academicAncestor(X,Z), advisor(Z,Y).
```

Clause

Head **Body**

Variable **Constant**

```
?- academicAncestor(X,"Fourier").
```

Query

UNIVERSITY OF CAMBRIDGE

QUEENS' COLLEGE CAMBRIDGE

Frustrating Datalog: programmer’s misery

```
lt100(X) :- X < 100.
?- in(X,1,50), lt100(X).
```

Frustrating Datalog: programmer's misery

```
pure(E) :- ! vars(E,_).
?- exp(E), pure(E).
```

Frustrating Datalog: programmer's misery

```
?- exp(E1), exp(E2), independent(E1,E2).

independent(E1,E2) :- pure(E1).
independent(E1,E2) :- pure(E2).
independent(E1,E2) :-
 vars(E1,V1), vars(E2,V2), ! intersect(V1,V2,_).
```

UNIVERSITY OF CAMBRIDGE

QUEENS' COLLEGE CAMBRIDGE

Frustrating Datalog: compiler writer's misery

```
?- exp(E₁), exp(E₂),
 (pure(E₁);
  pure(E₂);
  vars(E₁,V₁), vars(E₂,V₂), ! intersect(V₁,V₂,_)).
```

UNIVERSITY OF CAMBRIDGE

QUEENS' COLLEGE CAMBRIDGE

Overview

- Dataflow restrictions (well-moding & range restriction)
- Information flow (modes & adornment)
- Dataflow repair
- Synthesising control (reorder atoms)

Dataflow restrictions

Range restriction

'i' is not bound to a value.

```
2
3  int p(int i) { result = 42 }
```

```
p(I,42).
```

- Every variable in the head must appear in the body
- Why?
  - Domain independence
  - Finiteness
  - Termination
  - **Easy (not lazy) implementation**

Well-moding

'i' is not bound to a value.

```
predicate p(int i) { 1 <= 1 and i <= 10 }
predicate q(int i) { i in [1..10] }
```

```
p(I) :- I <= 1, I <= 10.
q(I) :- in(I,1,10).
```

- Not every predicate is created equal
- `<=` (as implemented in QL) is *extralogical;* `in` is *logical*
- Foreign predicates, parameters indexed in a database, semantically one way functions, etc.

UNIVERSITY OF CAMBRIDGE

QUEENS' COLLEGE CAMBRIDGE

Information flow

Mode = Binding requirement

- + means “needs to be bound”
- ? means “we don’t care”
- Static property independent of where the predicate occurs
- $\texttt{in}^{?++}(\texttt{I},\texttt{Lower},\texttt{Upper})$
  $\texttt{A} <=^{++} \texttt{B}$
  $\texttt{sha256}^{+?}(\texttt{Content},\texttt{Hash})$
  $\texttt{plus}^{++?,+?+,?++}(\texttt{A},\texttt{B},\texttt{C})$

Mode = Binding set (?)

- In QL, **binding sets** *seem* to be the equivalent notion
- `plus`$^{++?,+?+,?++}$`(A,B,C)`
- ```
  bindingset[a,b]
  bindingset[b,c]
  bindingset[a,c]
  predicate plus(int a, int b, int c) {
    a + b = c
  }
  ```
  ```

Adornment = active binding

- `b` for bound
- `f` for free
- Dynamic property depending on the context
- `?- p`$_{fb}$`(X,1), q`$_{bfb}$`(X,Y,0).`

Well-modable: Intuition

- `?- lt100(X).`
  ```
  lt100(X) :- X < 100.
  ```
- `?- in(X,1,50), lt100(X).`
  ```
  lt100(X) :- X < 100.
  ```

UNIVERSITY OF CAMBRIDGE

QUEENS' COLLEGE CAMBRIDGE

Well-modable: Formalisation

Need a b for every +

- $\texttt{?- lt100}_{\texttt{f}}\texttt{(X).}$
  $\texttt{lt100}_{\texttt{f}}\texttt{(X) :- X} <_{\texttt{fb}}^{\texttt{++}} \texttt{100.}$

- $\texttt{?- in}_{\texttt{fbb}}\texttt{(X,1,50), lt100}_{\texttt{b}}\texttt{(X).}$
  $\texttt{lt100}_{\texttt{b}}\texttt{(X) :- X} <_{\texttt{bb}}^{\texttt{++}} \texttt{100.}$

Well-modable vs well-moded

- Semantically clause heads and atoms are related

$$?\text{-}\ \mathtt{in}_{fbb}(X,1,50),\ \mathtt{lt100}_{b}(X).$$
$$\mathtt{lt100}_{b}(X)\ \text{:-}\ X <_{bb}^{++} 100.$$

- But the evaluator treats each clause independently

$$?\text{-}\ \mathtt{in}_{fbb}(X,1,50),\ \mathtt{lt100}_{b}(X).$$
$$\mathtt{lt100}_{f}(X)\ \text{:-}\ X <_{fb}^{++} 100.$$

- Well-modability is a **global** property, while well-modedness is a **local** one

Dataflow repair:
Transforming well-modable to well-moded and making clauses range-restricted

Repairing basic ill-modedness

- $?\text{-}\ \mathtt{in}_{fbb}(X,1,50),\ \mathtt{lt100}_{b}(X).$

  $\mathtt{lt100}_{f}(X)\ \text{:-}\ X <_{fb}^{++} 100.$

- $?\text{-}\ \mathtt{in}_{fbb}(X,1,50),\ \mathtt{lt100}_{b}(X).$

  $\mathtt{lt100}_{f}(X)\ \text{:-}$

  $\quad \mathtt{in}_{fbb}(X,1,50),\ X <_{bb}^{++} 100.$

UNIVERSITY OF CAMBRIDGE

QUEENS' COLLEGE CAMBRIDGE

Repairing basic range-restriction

- ?- exp(E_1), exp(E_2), independent(E_1,E_2).

  independent(E_1,E_2) :- pure(E_1).
  independent(E_1,E_2) :- pure(E_2).
  …

- ?- exp(E_1), exp(E_2), independent(E_1,E_2).

  independent(E_1,E_2) :- exp(E_2), pure(E_1).
  independent(E_1,E_2) :- exp(E_1), pure(E_2).
  …

UNIVERSITY OF CAMBRIDGE

QUEENS' COLLEGE CAMBRIDGE

Not so basic dataflow 1

- ```
  ?- pClo(X,1).
  pClo(X,Y) :- p?+(X,Y).
  pClo(X,Z) :- pClo(X,Y), p?+(Y,Z).
  ```
- Is this well-moded?
  ```

Not so basic dataflow 2

- `?- p(X,1), r(Y), p(X,Y).`
  `p(X,Y) :- q(X).`
- What flows into Y of p?

What are we trying to find?

- For well-modedness, target is body **atom parameters**. What flows into `X` of `X <++ 100`?

```
?- in(X,1,50), lt100(X).
lt100(X) :- X <++ 100.
```

- For range restriction, target is **predicate parameters**. What flows into second parameter of `independent`?

?- exp(E_1), exp(E_2), independent(E_1,E_2).
independent(E_1,E_2) :- pure(E_1).

What are we **really** trying to find?

- Enumerable domains for variables!
- QL makes this much easier with its **characteristic predicates**
- In pure Datalog, we must look hard for them

How to find these domains? (sketch)

1. Extract all local flow into a graph

2. Compute the shortest paths
   from data sources (constants, positive atoms)
   to data sinks (offending atom/predicate parameters)

3. Ensure data paths to offending sinks are **closed**

4. Extract domain into new predicate and prepend it to the problematic clause body

Implementation

- Feature extraction can be done in Datalog via MetaDL
- Flow path computation is about ~15 pure Datalog clauses
- Transformation cannot (yet) be done in Datalog!
- Similar analysis required for Magic Set Rewriting, potential infrastructure reuse

Mode inference and atom reordering

Adornments are contextual

Reminder: need a b for every +

- $\texttt{auth}_{\texttt{f}}\texttt{(U) :- hash}_{\texttt{ff}}^{+?}\texttt{(P,H), password}_{\texttt{bf}}\texttt{(U,P), valid}_{\texttt{bb}}\texttt{(U,H).}$ ❌
- $\texttt{auth}_{\texttt{f}}\texttt{(U) :- password}_{\texttt{ff}}\texttt{(U,P), hash}_{\texttt{bf}}^{+?}\texttt{(P,H), valid}_{\texttt{bb}}\texttt{(U,H).}$ ✅

Naïvely enumerate orderings?

- $\mathtt{auth_f(U)}$:- $\mathtt{hash_{ff}^{+?}(P,H)}$, $\mathtt{password_{bf}(U,P)}$, $\mathtt{valid_{bb}(U,H)}$.❌
- $\mathtt{auth_f(U)}$:- $\mathtt{hash_{ff}^{+?}(P,H)}$, $\mathtt{valid_{fb}(U,H)}$, $\mathtt{password_{bb}(U,P)}$.❌
- $\mathtt{auth_f(U)}$:- $\mathtt{password_{ff}(U,P)}$, $\mathtt{hash_{bf}^{+?}(P,H)}$, $\mathtt{valid_{bb}(U,H)}$.✅
- $\mathtt{auth_f(U)}$:- $\mathtt{password_{ff}(U,P)}$, $\mathtt{valid_{bf}(U,H)}$, $\mathtt{hash_{bb}^{+?}(P,H)}$.✅
- $\mathtt{auth_f(U)}$:- $\mathtt{valid_{ff}(U,H)}$, $\mathtt{password_{bf}(U,P)}$, $\mathtt{hash_{bb}^{+?}(P,H)}$.✅
- $\mathtt{auth_f(U)}$:- $\mathtt{valid_{ff}(U,H)}$, $\mathtt{hash_{fb}^{+?}(P,H)}$, $\mathtt{password_{bb}(U,P)}$.✅

It's a global reordering problem

- $?-\ auth_{f}(U)$
  $auth_{f}(U)\ :-\ check_{ff}(U,P),\ password_{bb}(U,P).$
  $check_{ff}(U,P)\ :-\ hash_{ff}^{+?}(P,H),\ valid_{bb}(U,H).$ ❌

- $?-\ auth_{f}(U)$
  $auth_{f}(U)\ :-\ password_{ff}(U,P),\ check_{bb}(U,P).$
  $check_{bb}(U,P)\ :-\ hash_{bf}^{+?}(P,H),\ valid_{bb}(U,H).$ ✅

- Combinatorial explosion! Naïve won't do.

What are we really trying to compute?

- Orderings, but also a summary of binding requirements
- Mode inference for derived predicates
- What should the ideal mode of `check` is?

$$\mathtt{check}_{bb}(\mathtt{U},\mathtt{P})\ \mathtt{:-}\ \mathtt{hash}_{bf}^{+?}(\mathtt{P},\mathtt{H}),\ \mathtt{valid}_{bb}(\mathtt{U},\mathtt{H}).$$

- `+?` or `??`? `??` is strictly better
- We have submoding! Only want the most general mode
- Each mode is coupled to orderings that lead to it

Multiple modes due to ordering

- ```
  clientCheck(P) :- weak(P,H).
  serverCheck(H) :- weak(P,H).
  weak(P,H) :- hash+?(P,H), rainbow+?(H,P).
  ```
- The mode of weak is `+?,?+`
  ```

Mode inference algorithm (sketch)

1. Start with input modes, assume others are pure
2. For each clause, construct a greedy scheduling graph
3. Find the most relaxed modes and their orderings
4. Combine modes of clauses sharing head predicates
5. Go to (2) repeat until convergence

Two heuristics against naïveté

- Pure atoms can always be scheduled first in any order
- Scheduling might make other atoms pure
- ?- ● $p^{+}(Z)$, $q^{+?}(X,Z)$, $r^{??}(X,Y)$.
  ?- $r^{??}(X,Y)$, ● $p^{+}(Z)$, $q^{??}(X,Z)$.
  ?- $r^{??}(X,Y)$, $q^{??}(X,Z)$, ● $p^{?}(Z)$.
  ?- $r^{??}(X,Y)$, $q^{??}(X,Z)$, $p^{?}(Z)$ ●.

Properties

- Sound and complete
- Incremental
- Highly parallelisable
- Closely mimics properties of Datalog evaluation
- Still exponential in degenerate case but only locally
  ```
  ?- p+(X), q+(Y), r+(Z).
  ```

UNIVERSITY OF CAMBRIDGE

QUEENS' COLLEGE CAMBRIDGE

Conclusion

- Well-modedness and range restriction are sources of "is not bound to a value” errors
- They can be fixed via program transformations
- Dataflow restricted atoms can be used as if purely logical
- Modes (or binding sets) can always be inferred
- Datalog analysis is nice because Datalog is nice!

Thanks. Questions?

Mistral.Contrastin@cl.cam.ac.uk
dodisturb.me
@madgen_