Functions as types or the “Hoare logic” of functional dependencies

Author: José Nuno Oliveira

Speaker: Ricardo Gonçalves

Outline

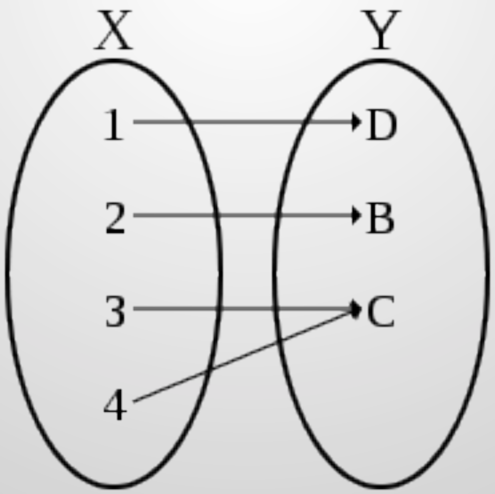
- Funtional Dependency (DBs)
- Injectivity
- Hoare Logic (Pre/Post Conditions)
- Aplication 1: Type Checking
- Aplication 2: Query Optimization
- Conclusion & Future work

Introdution

- Unify Functional Dependency and Hoare Logic theories, by encoding them in abstract algebra;
- Using this general theory, can we type check database programming?
- Also can we do Query Optimization?
- What are the consequences..

Functional Dependency (1/3)

StudentID	Semester	Lecture	TA
1234	6	Numerical Methods	John
2380	4	Numerical Methods	Peter
1234	6	Visual Computing	Amina
1201	4	Numerical Methods	Peter
1201	4	Physics II	Simone



Functional Dependency (2/3)

- Logical definition:

$$\forall\ t, t' :\ t, t' \in T\ \Rightarrow\ (t[x] = t'[x] \Rightarrow t[y] = t'[y])$$

- Applying relational algebra rules, we obtain:

$$[\![T]\!] \cdot x^\circ \cdot x \cdot [\![T]\!]^\circ \subseteq y^\circ \cdot y$$

Where [[T]] is the binary relation of T

Functional Dependency (3/3)

- Let's generalized table T to an arbitrary relation R:

$$R \cdot f^\circ \cdot f \cdot R^\circ \subseteq g^\circ \cdot g$$

- Informally, every unique f via R has an unique g
- So let's represent this by:

$$f \xrightarrow{R} g$$

Injectivity

- Let's define injectivity by ≤ using the kernel:

$$R \leq S \quad \triangleq \quad \text{ker } S \subseteq \text{ker } R$$

- In a sense, the bigger the kernel, the less injective it is.
- For a total function, the kernel is bounded by:

$$! \leq R \leq id$$

Hoare Logic

- A Program R, a Pre-condition p and post-condition q are represented as:

$$\{p\}R\{q\} \quad == \quad p \xrightarrow{R_1} q$$

- We can define this relation in our algebric notation using injectivity:

$$\{p\}R\{q\} \equiv q \leq p \cdot R^\circ$$

Type Checking a DB (1/2)

- We want to know what it means for the merging of two database files to satisfy a particular functional dependency:

$$g \xleftarrow{R \cup S} f$$

Type Checking a DB (2/2)

$$g \xleftarrow{R \cup S} f$$

$\equiv$ { definition (13) ; converse distributes by union }

$$g \leq f \cdot (R^\circ \cup S^\circ)$$

$\equiv$ { relational composition distributes through union }

$$g \leq f \cdot R^\circ \cup f \cdot S^\circ$$

$\equiv$ { algebra of injectivity (20); definition (13) again, twice }

$$g \xleftarrow{R} f \quad \wedge \quad g \xleftarrow{S} f \quad \wedge \quad R \cdot \mathsf{ker}\ f \cdot S^\circ \subseteq \mathsf{ker}\ g$$

$\equiv$ { introduce "mutual dependency" shorthand }

$$g \xleftarrow{R} f \quad \wedge \quad g \xleftarrow{S} f \quad \wedge \quad g \xleftarrow{R,S} f$$

Query Optimization (1/3)

- Let's have a DB table Movies(Title,Director,Actor)

$$\{(d, a') \mid t = t', (t, d, a) \in Movies, (t', d', a') \in Movies\}$$

- Which in linear algebra is defined as (abbreviated types):

$$d \cdot M \cdot (\mathsf{ker}\ t) \cdot M \cdot a^{\circ} = X$$

- The aim is to obtain a solution X containing only one instance of M.

Query Optimization (2/3)

$$d \xleftarrow{M} t$$

$\equiv \quad \{ \ (13) \ \}$

$$d \leq t \cdot M^\circ$$

$\equiv \quad \{ \ \text{expanding (11,12);}\ M^\circ = M \text{ since } M \text{ is a set} \ \}$

$$M \cdot t^\circ \cdot t \cdot M \subseteq d^\circ \cdot d$$

$\equiv \quad \{ \ \text{composition } (\cdot M) \text{ with a set (partial identity) is a closure operator} \ \}$

$$M \cdot t^\circ \cdot t \cdot M \subseteq d^\circ \cdot d \cdot M$$

$\Rightarrow \quad \{ \ \text{shunting (16,17); monotonicity of } (\cdot a^\circ)\text{; kernel (11)} \ \}$

$$d \cdot M \cdot (\mathsf{ker}\ t) \cdot M \cdot a^\circ \subseteq d \cdot M \cdot a^\circ$$

Query Optimization (3/3)

$$
\begin{array}{ll}
& d \cdot M \cdot a^\circ \subseteq d \cdot M \cdot (\mathsf{ker}\ t) \cdot M \cdot a^\circ \\
\Leftarrow & \quad \{\ id \subseteq \mathsf{ker}\ t \text{ because kernels are equivalence relations } \} \\
& d \cdot M \cdot a^\circ \subseteq d \cdot M \cdot M \cdot a^\circ \\
\equiv & \quad \{\ M \cdot M = M \cap M = M \text{ because } M \text{ is a set } \} \\
& d \cdot M \cdot a^\circ \subseteq d \cdot M \cdot a^\circ
\end{array}
$$

- So we know our X, let’s revert back to the prior notation (with variables):

$$X \ = \ \{(d, a') \mid (t, d, a') \in Movies\}$$

Conclusion/Future Work

- Concept prove of unifying theories through LA
- We can adapt/generalize software from on side to the other (Prover9)
- Type Checking and Query Optimization are "automated" through LA
- Another possible theory to adapt is the Strongest Invariant for loops