Dynamic Invariant Detection for Relational Databases

Jake Cobb[1], Gregory M. Kapfhammer[2], James A. Jones[3], Mary Jean Harrold[4]

[1]Georgia Institute of Technology
[2]Allegheny College
[3]University of California, Irvine

WODA 2011 – July 18, 2011

Outline

Dynamic Invariants

Definition

A dynamic invariant is a property that is observed to hold during a *series of executions*.

- Not guaranteed for all possible executions.
- May reflect property of:
  - Program
  - Inputs

Daikon

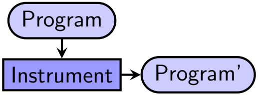
Daikon [Ernst et al. 2001] is a dynamic invariant detection engine.

- Collect data traces for variables at *program point*s.
- Compare to pool of potential invariants.
- Output remaining invariants that meet confidence threshold.

Daikon

Daikon [Ernst et al. 2001] is a dynamic invariant detection engine.

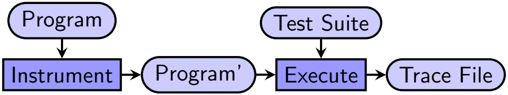
- Collect data traces for variables at *program points*.
- Compare to pool of potential invariants.
- Output remaining invariants that meet confidence threshold.



Daikon

Daikon [Ernst et al. 2001] is a dynamic invariant detection engine.

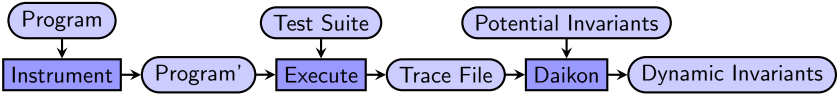
- Collect data traces for variables at *program points*.
- Compare to pool of potential invariants.
- Output remaining invariants that meet confidence threshold.



Daikon

Daikon [Ernst et al. 2001] is a dynamic invariant detection engine.

- Collect data traces for variables at *program points*.
- Compare to pool of potential invariants.
- Output remaining invariants that meet confidence threshold.



Daikon

Many applications of dynamic invariants in software engineering:

- Programmer understanding
- Run-time checking
- Integration testing
- Interface discovery
- Test-input generation
- ...

Relational Databases

Relational Model

TableA		
ColumnA	**ColumnB**	...
1	'Data'	...
2	'Values'	...
...		

TableB		
ColumnC	**ColumnD**	...
...		

SQL

SQL (Structured Query Language) is a standard and query language for relational database management systems (RDBMS).

SQL

SQL (Structured Query Language) is a standard and query language for relational database management systems (RDBMS).

Data Definition

A schema is a collection of table definitions.

```
CREATE TABLE person (
  id   INT,
  name VARCHAR(100) NOT NULL,
  age  INT(3),
  PRIMARY KEY (id)
)
```

SQL

SQL (Structured Query Language) is a standard and query language for relational database management systems (RDBMS).

Data Definition

A schema is a collection of table definitions.

```
CREATE TABLE person (
  id   INT,
  name VARCHAR(100) NOT NULL,
  age  INT(3),
  PRIMARY KEY (id)
)
```

Create, Read, Update and Delete (CRUD) Operations

```
INSERT INTO person (id, name, age) VALUES (1, 'John', 38)
SELECT name FROM person WHERE age >= 30 AND age <= 40
UPDATE person SET name = 'Jan' WHERE id = 2
DELETE FROM person WHERE id = 2
```

Outline

Structural Mapping

Program Element	**DB Element**
Program Point	Table
Variable	Column
Occurence	Row

Structural Mapping

Program Element	**DB Element**
Program Point	Table
Variable	Column
Occurence	Row

Detect invariants for:

- Individual columns.
- Between columns in a given row.

Structural Mapping

Program Element	**DB Element**
Program Point	Table
Variable	Column
Occurence	Row

Detect invariants for:

- Individual columns.
- Between columns in a given row.

Example

id	**name**	**age**	**employed**	...
1	'John Smith'	38	5	...
2	'Jan Downing'	22	2	...

Data Mapping

Daikon Concepts

- Representation type
  - `int`
  - `double`
  - `String`
  - `int[]`
- Comparability

Data Mapping

Group	Name	SQL Types	Java Type

1	Text	`CHAR`
		`VARCHAR`
		`TEXT`
			`String`

2	Integer	`INTEGER`
		`NUMERIC`
		`BIT`
			`int`

3	Decimal	`FLOAT`
		`DOUBLE`
		`REAL`
		`DECIMAL`
			`double`

4	Binary	`BLOB` `BIT`	`byte[]`
5	Text Set	`SET`	`String[]`
6	Datetime	`DATETIME` `TIMESTAMP`	`String`
7	Date	`DATE`	`String`
8	Time	`TIME`	`String`
9	Interval	`INTERVAL`	`int`
10	Primary Key	`INTEGER`	*reference*

Data Mapping

`NULL` Values

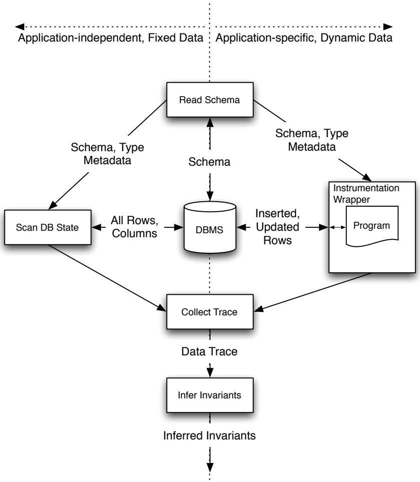
- `NULL` is a possible value for any SQL type.
- Daikon does not accept `null` for primitive representation types, e.g. `int`.

Data Mapping

`NULL` Values

- `NULL` is a possible value for any SQL type.
- Daikon does not accept `null` for primitive representation types, e.g. `int`.
- Introduce synthetic variable for each `NULL`-able column.
  - Representation type is `hashcode` (*reference*).
  - Value is either `null` or a constant.

Process Overview



Implementation

Trace Collector

- Python[1] program:
  - Input: DB connection information.
  - Output: Daikon declarations and data trace files.
- Process:
  1. Read schema metadata to determine tables, columns and data mapping.
  2. Write declarations file and serialize mapping info for reuse.
  3. `SELECT` table contents, transform data by mapping, write to GZip'd trace file.
- Supports various RDBMS via SQLAlchemy.

[1] ...plus a tiny bit of Cython

Implementation

Instrumentation Wrapper

- Modified P6Spy JDBC driver wrapper.
- On connection, capture information and initiate initial metadata read and trace.
- On statement execution, append trace if data could be modified.
  - `INSERT` statement.
  - `UPDATE` statement.
  - Unknown (e.g. a stored procedure call.)
  - Ignore others, including `DELETE` and `TRUNCATE`.

Outline

Subjects

Fixed Data Sets

Subject	Tables	Columns	Rows
`world`	3	24	5302
`sakila`	23	131	50,086
`menagerie`	2	10	19
`employees`	6	24	3,919,015

- MySQL sample databases for training, certification and testing.
- Trace entire dataset.

Subjects

Database Applications

Program	iTrust	JWhoisServer	JTrac
Tables	30	7	13
Columns	177	57	126
KLOC	25.5 (Java), 8.6 (JSP)	6.7	12
Test Cases	787	67	41

- Java applications driven by a database.
- Wrap real DB driver in a modified P6Spy driver.
- Execute the test suite.

Invariant Quality

Meaningful Invariants

Invariants that capture a semantic relationship.

Invariant Quality

Meaningful Invariants

Invariants that capture a semantic relationship.

- `dept_emp.from_date <= dept_emp.to_date`
- `employees.gender one of { "F", "M" }`
- `employees.birth_date < employees.hire_date`
- `country.Population >= 0`
- `icdcodes.Chronic one of { "no", "yes" }`

Spurious Invariants

Spurious Invariants

- Vacuous invariants reflect a meaningless relationship.
- Lack-of-data invariants result from limited data samples.

Spurious Invariants

Spurious Invariants

- Vacuous invariants reflect a meaningless relationship.
  - `patients.phone1 <= patients.BloodType`
  - `patients.lastName >= patients.address1`
  - `cptcodes.Description != cptcodes.Attribute`
- Lack-of-data invariants result from limited data samples.

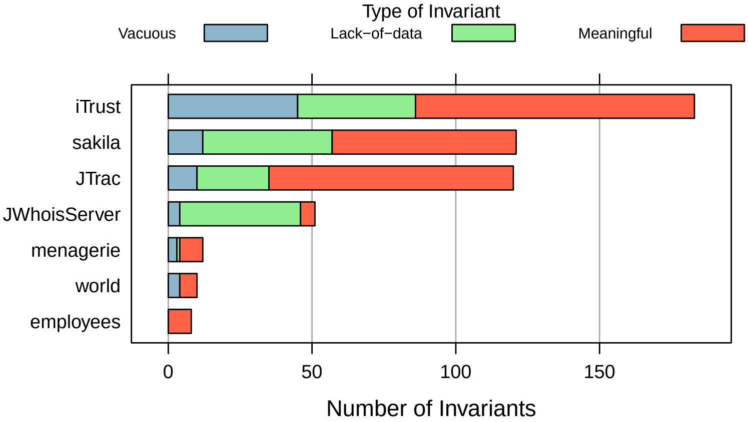
Spurious Invariants

Spurious Invariants

- Vacuous invariants reflect a meaningless relationship.
  - `patients.phone1 <= patients.BloodType`
  - `patients.lastName >= patients.address1`
  - `cptcodes.Description != cptcodes.Attribute`
- Lack-of-data invariants result from limited data samples.
  - `mntnr.login == "mntnt"`
  - `inetnum.changed == "2006-10-14 16:21:09"`
  - `person.name one of { "no name company", "persona non grata"}`

Invariant Quality

Results



Schema Modification

Schema Modification

- Some invariants can be enforced by the schema definition.
- Schema enforcement provides a stronger assurance of data integrity than application enforcement.
- Analyze enforceable invariants:
  - Already enforced by the schema.
  - Suggest modification to enforce the invariant.

Schema Modification

Schema Enforced

Invariant	Schema Definition
`employees.gender one of { "F", "M" }`	`ENUM('F','M')`
`countrylanguage.IsOfficial one of { "F", "T" }`	`ENUM('F','T')`
`customer.active one of { 0, 1 }`	`TINYINT(1)`
`inventory.film_id >= 1`	`SMALLINT(5) UNSIGNED`
`spaces.guest_allowed one of { 0, 1 }`	`BIT(1)`

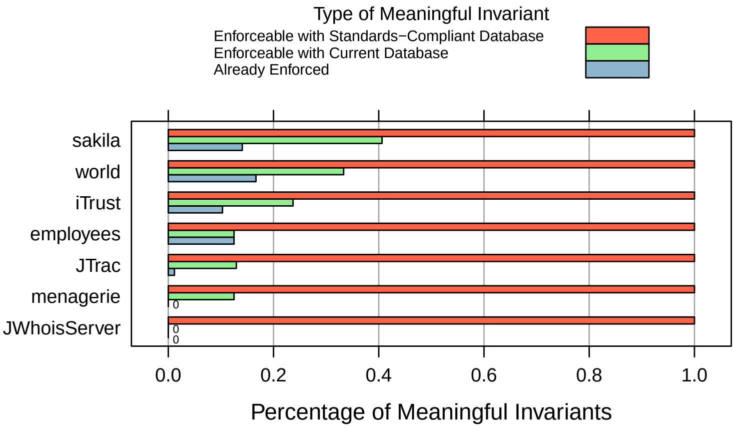
Schema Modification

Schema Enforceable

Invariant	Schema	Modification
`isnull(message.message) != null`	`TEXT`	`NOT NULL`
`isnull(film_text.description) != null`	`TEXT`	`NOT NULL`
`isnull(history.time_stamp) != null`	`DATETIME`	`NOT NULL`
`user_space_roles.user_id >= 1`	`BIGINT(20)`	`UNSIGNED`
`pet.sex one of { "f", "m" }`	`CHAR(1)`	`ENUM('m','f')`
`country.Population >= 0`	`INT(11)`	`UNSIGNED`
`isnull(titles.to_date) != null`	`DATE`	`NOT NULL`

Schema Modification

Results



Conclusions and Future Work

Conclusions

- Meaningful invariants may be mined from databases and database applications.
- Invariant quality depends on diverse data.
- Data integrity may be enhanced by using invariants for schema modification.

Conclusions and Future Work

Conclusions

- Meaningful invariants may be mined from databases and database applications.
- Invariant quality depends on diverse data.
- Data integrity may be enhanced by using invariants for schema modification.

Future Work

- Invariants between multiple tables.
- Invariants for individual queries.
- Explore additional client applications.

Questions

Dynamic Invariant Detection for
Relational Databases

Thank you for your time and attention.

Questions?

Georgia Institute of Technology

Allegheny College
Meadville, Pennsylvania

UC Irvine | University of California