A Family of Test Adequacy Criteria for Database-Driven Applications

Gregory M. Kapfhammer
Department of Computer Science
University of Pittsburgh

Department of Computer Science
Allegheny College

Mary Lou Soffa
Department of Computer Science
University of Pittsburgh

Motivation

The Risks Digest, Volume 22, Issue 64, 2003

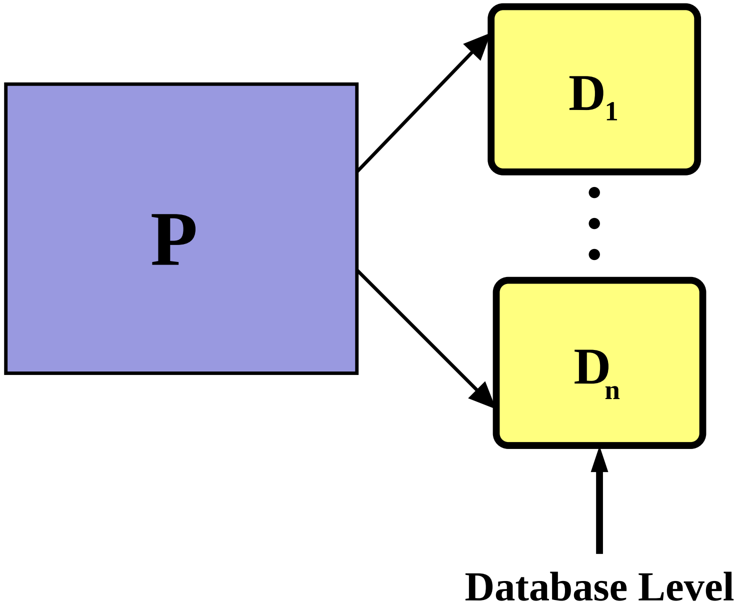
Jeppesen reports airspace boundary problems

About 350 airspace boundaries contained in Jeppesen NavData are incorrect, the FAA has warned. The error occurred at Jeppesen after a software upgrade when information was pulled from a database containing 20,000 airspace boundaries worldwide for the March NavData update, which takes effect March 20.

Testing Challenges

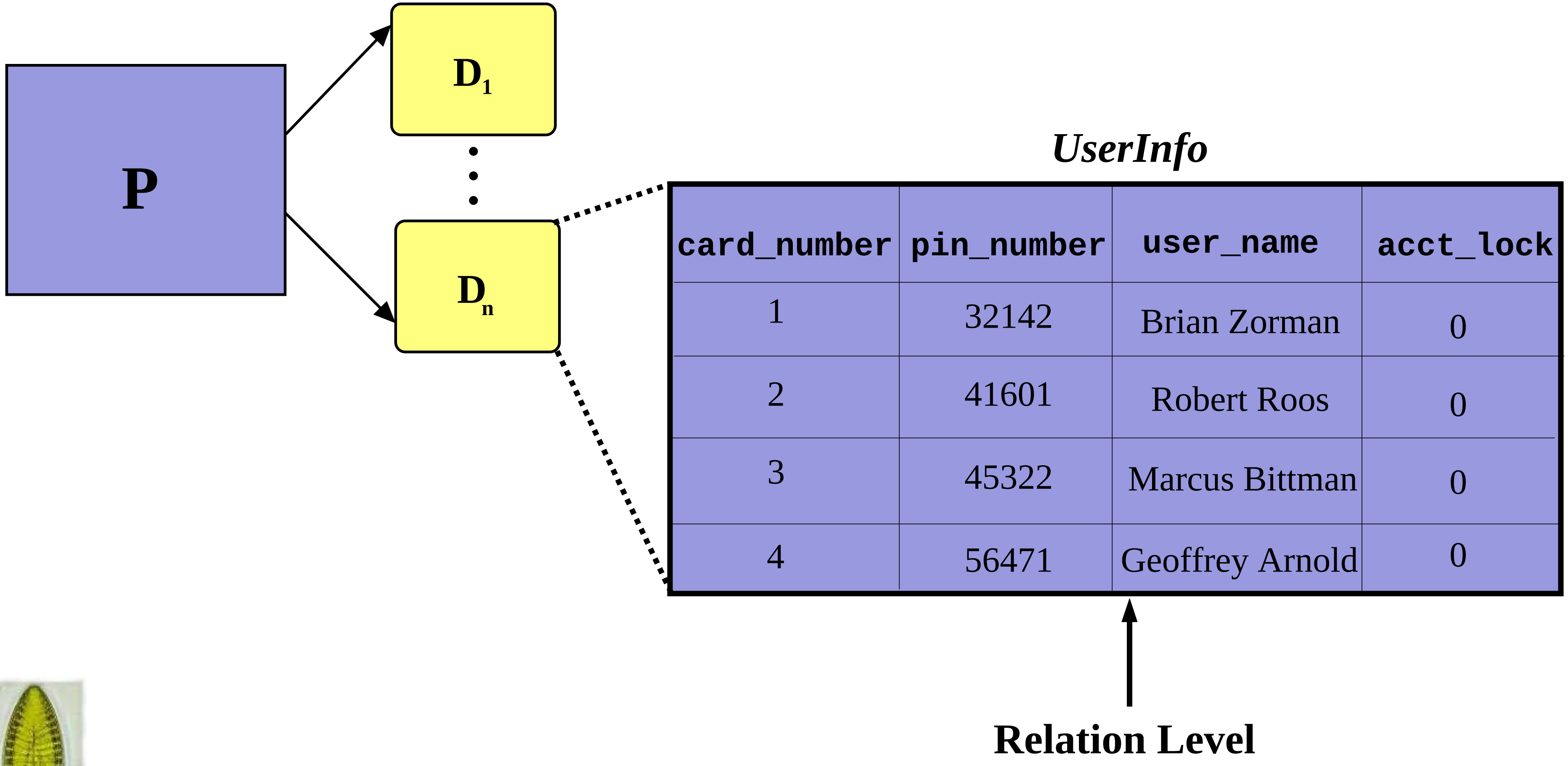
- Should consider the environment in which software applications execute
- Must test a program and its interaction with a database
- Database-driven application’s state space is well-structured, but infinite (Chays et al.)
- Need to show program does not violate database integrity, where *integrity* = *consistency* + *validity* (Motro)
- Must locate program and database coupling points that vary in granularity

Database-Driven Applications



- A program can interact with a database at different levels of granularity

Database-Driven Applications

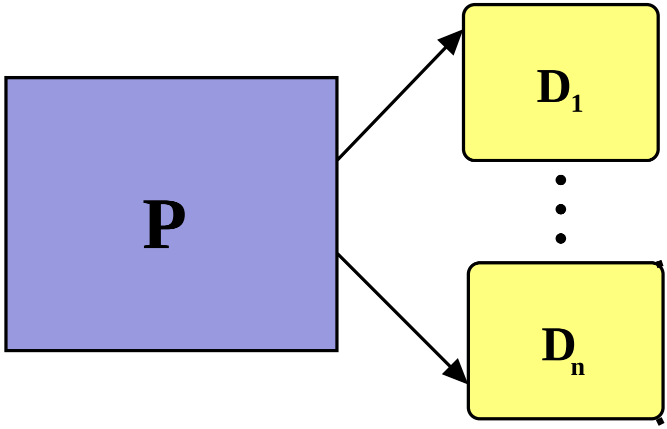


UserInfo

card_number	pin_number	user_name	acct_lock
1	32142	Brian Zorman	0
2	41601	Robert Roos	0
3	45322	Marcus Bittman	0
4	56471	Geoffrey Arnold	0

Relation Level

Database-Driven Applications

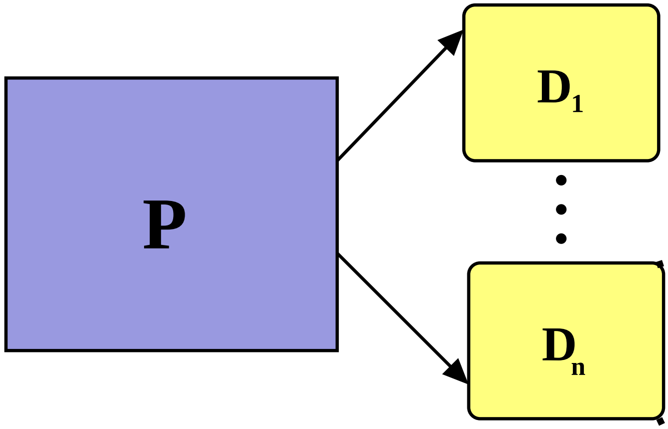


UserInfo

card_number	pin_number	user_name	acct_lock
1	32142	Brian Zorman	0
2	41601	Robert Roos	0
3	45322	Marcus Bittman	0
4	56471	Geoffrey Arnold	0

Record Level

Database-Driven Applications

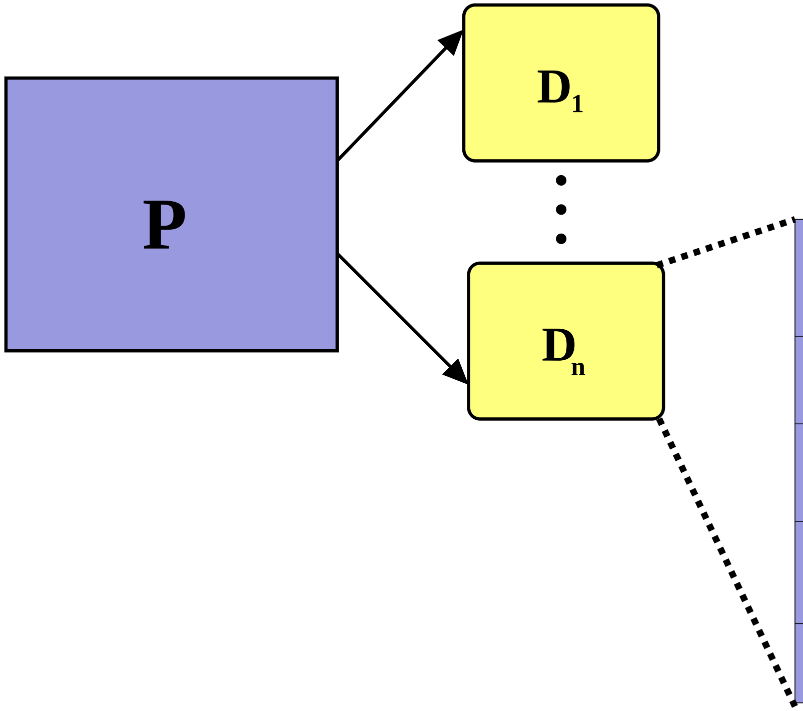


UserInfo

card_number	pin_number	user_name	acct_lock
1	32142	Brian Zorman	0
2	41601	Robert Roos	0
3	45322	Marcus Bittman	0
4	56471	Geoffrey Arnold	0

Attribute Level

Database-Driven Applications



UserInfo

card_number	pin_number	user_name	acct_lock
1	32142	Brian Zorman	0
2	41601	Robert Roos	0
3	45322	Marcus Bittman	0
4	56471	Geoffrey Arnold	0

Attribute Value Level

Test Adequacy Criteria

- P violates a database D_i's validity when it:
  - **(1-v)** inserts entities into D_i that do not reflect real world
- P violates a database D_i's completeness when it:
  - **(1-c)** deletes entities from D_i that still reflect real world
- In order to verify **(1-v)** and **(1-c)**, T must cause P to define and then use entities within $D_1, \ldots, D_n$!

Data Flow Information

- Interaction point: `‘‘UPDATE UserInfo SET acct_lock=1’’ + ‘‘WHERE card_number=’’ + c_n + ‘‘;’’;`
    - Database Level: *define(BankDB)*
    - Attribute Level: *define(acct_lock)* and *use(card_number)*
- Data flow information varies with respect to the granularity of the database interaction

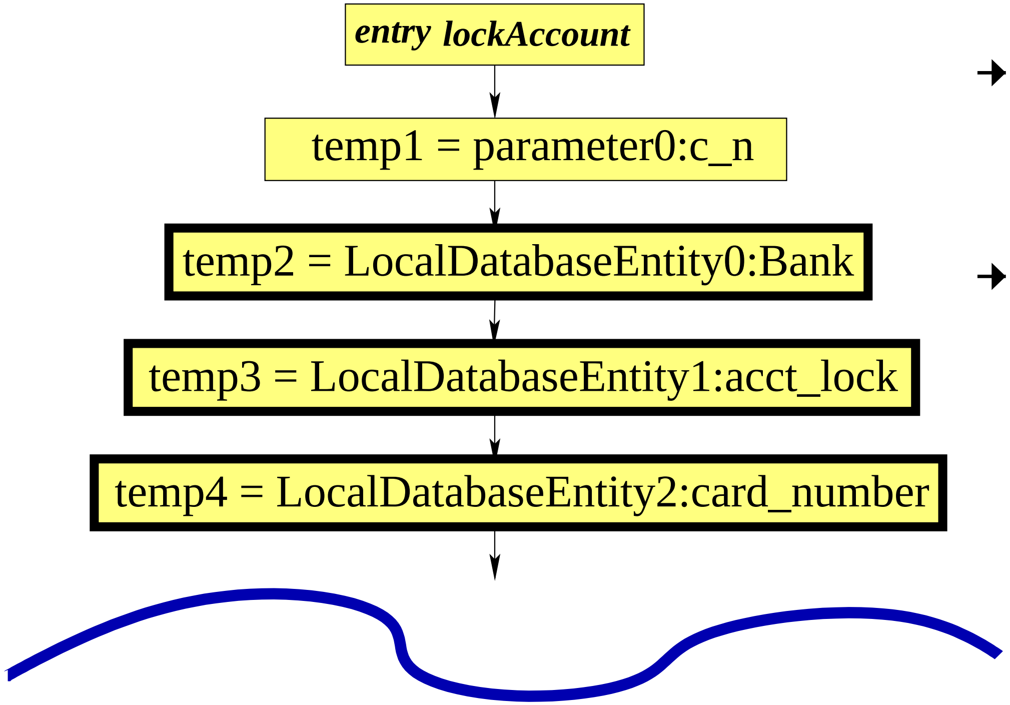
Database Entities

UserInfo

card_number	pin_number	user_name	acct_lock
1	32142	Brian Zorman	0
2	41601	Robert Roos	0
3	45322	Marcus Bittman	0
4	56471	Geoffrey Arnold	0

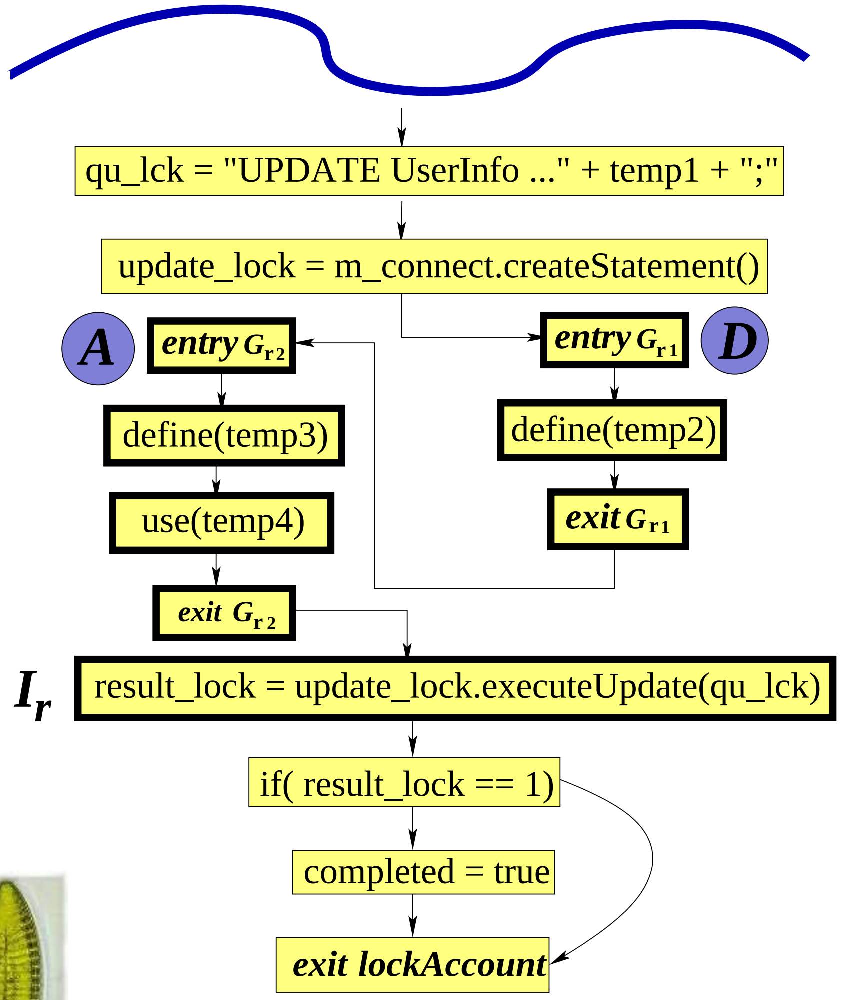
$$A_v(I_r) = \{ \boxed{1} , \boxed{32142} , \ldots , \boxed{\text{Geoffrey Arnold}} , \boxed{0} \}$$

The DICFG: A Unified Representation


entry lockAccount
temp1 = parameter0:c_n
temp2 = LocalDatabaseEntity0:Bank
temp3 = LocalDatabaseEntity1:acct_lock
temp4 = LocalDatabaseEntity2:card_number

- “Database-enhanced” CFG for `lockAccount`
- Define temporaries to represent the program’s interaction at the levels of database and attribute

The DICFG: A Unified Representation



qu_lck = "UPDATE UserInfo ..." + temp1 + ";"

update_lock = m_connect.createStatement()

A

entry G_{r2}

define(temp3)

use(temp4)

exit G_{r2}

entry G_{r1}

D

define(temp2)

exit G_{r1}

I_r

result_lock = update_lock.executeUpdate(qu_lck)

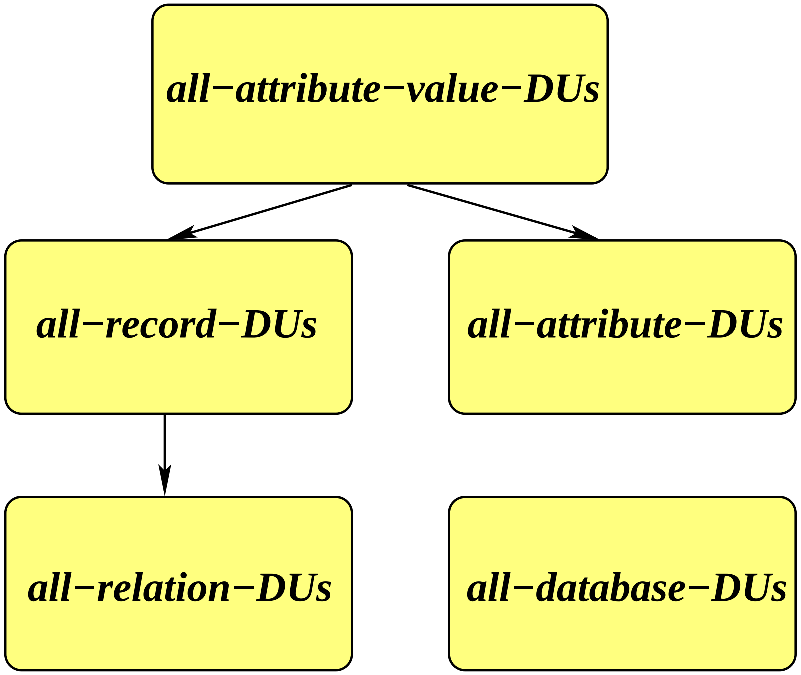
if(result_lock == 1)

completed = true

exit lockAccount

- Database interaction graphs (DIGs) are placed before the database interaction point I_r
- Multiple DIGs can be integrated into a single CFG

Test Adequacy Criteria



all−attribute−value−DUs

all−record−DUs

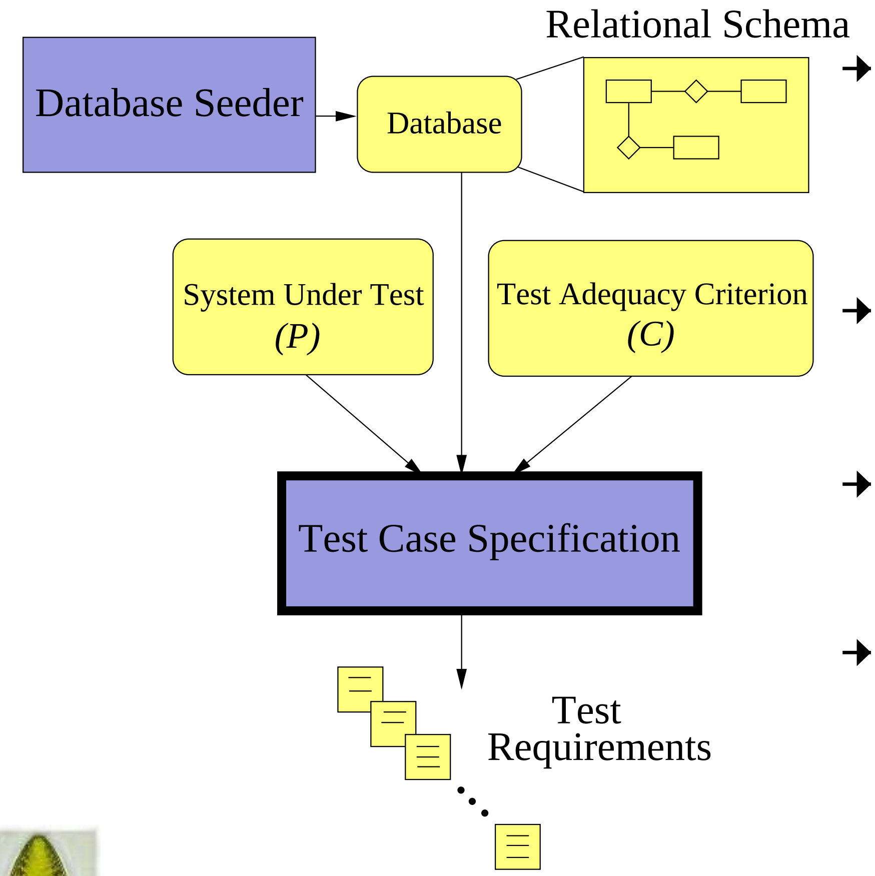
all−attribute−DUs

all−relation−DUs

all−database−DUs

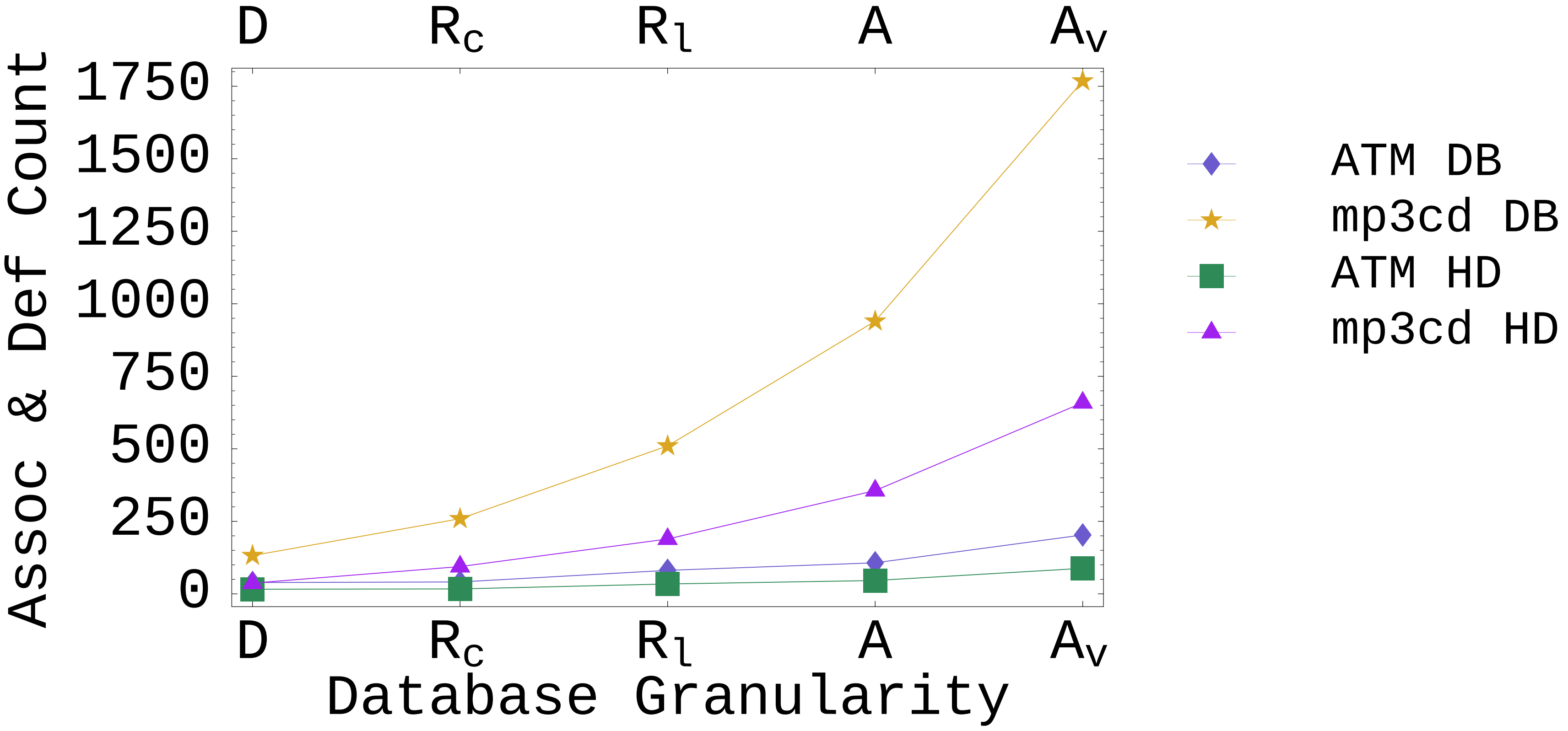
- Database interaction association (DIA) involves the *def* and *use* of a database entity
- DIAs can be located in the DICFG with data flow analysis
- *all-database-DUs* requires tests to exercise all DIAs for all of the accessed databases

Generating Test Requirements



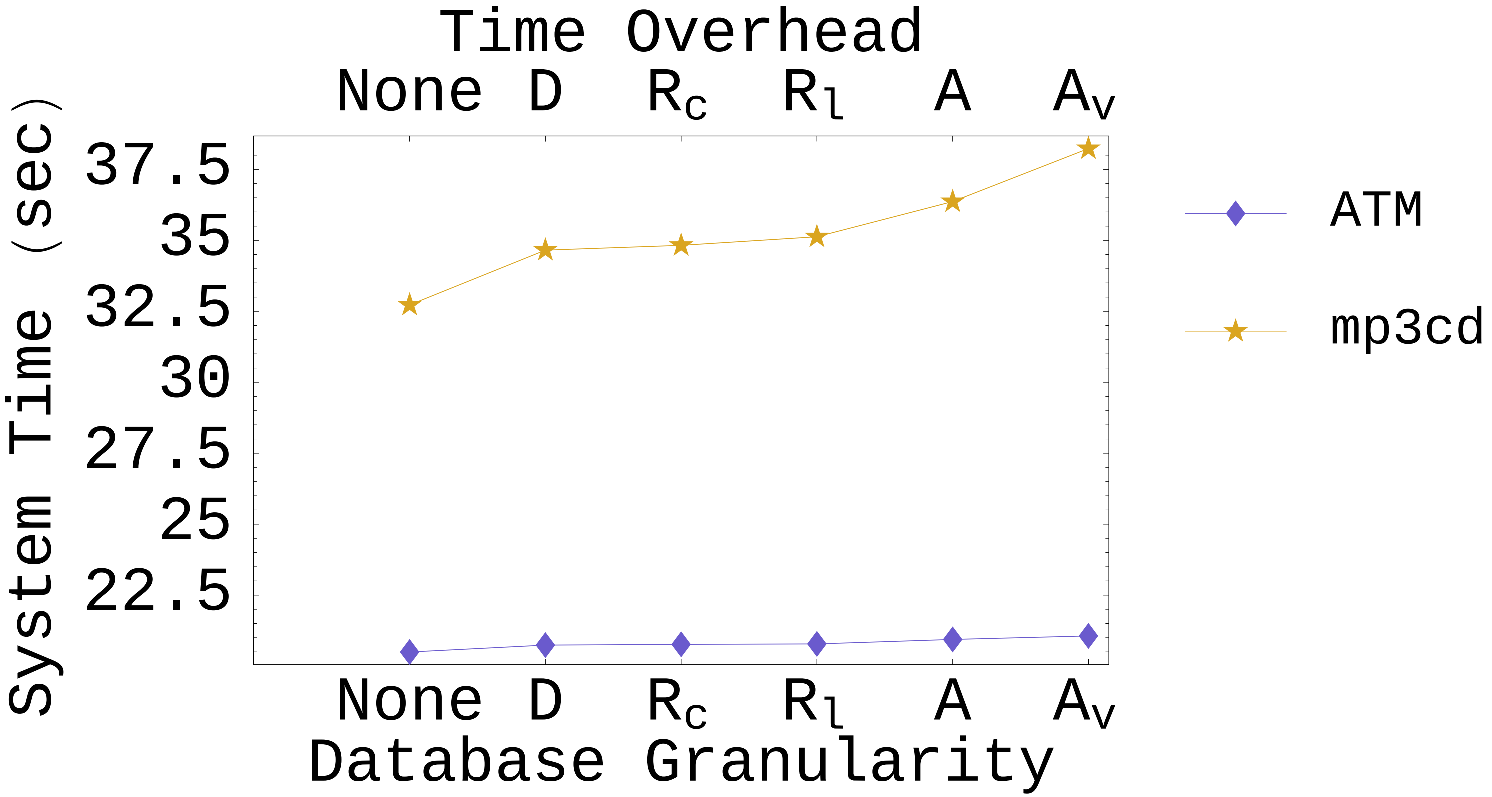
- Measured time and space overhead when computing family of test adequacy criteria
- Modified `ATM` and `mp3cd` to contain appropriate database interaction points
- Soot 1.2.5 to calculate intraprocedural associations
- GNU/Linux workstation with kernel 2.4.18-smp and dual 1 GHz Pentium III Xeon processors

Counting Associations and Definitions



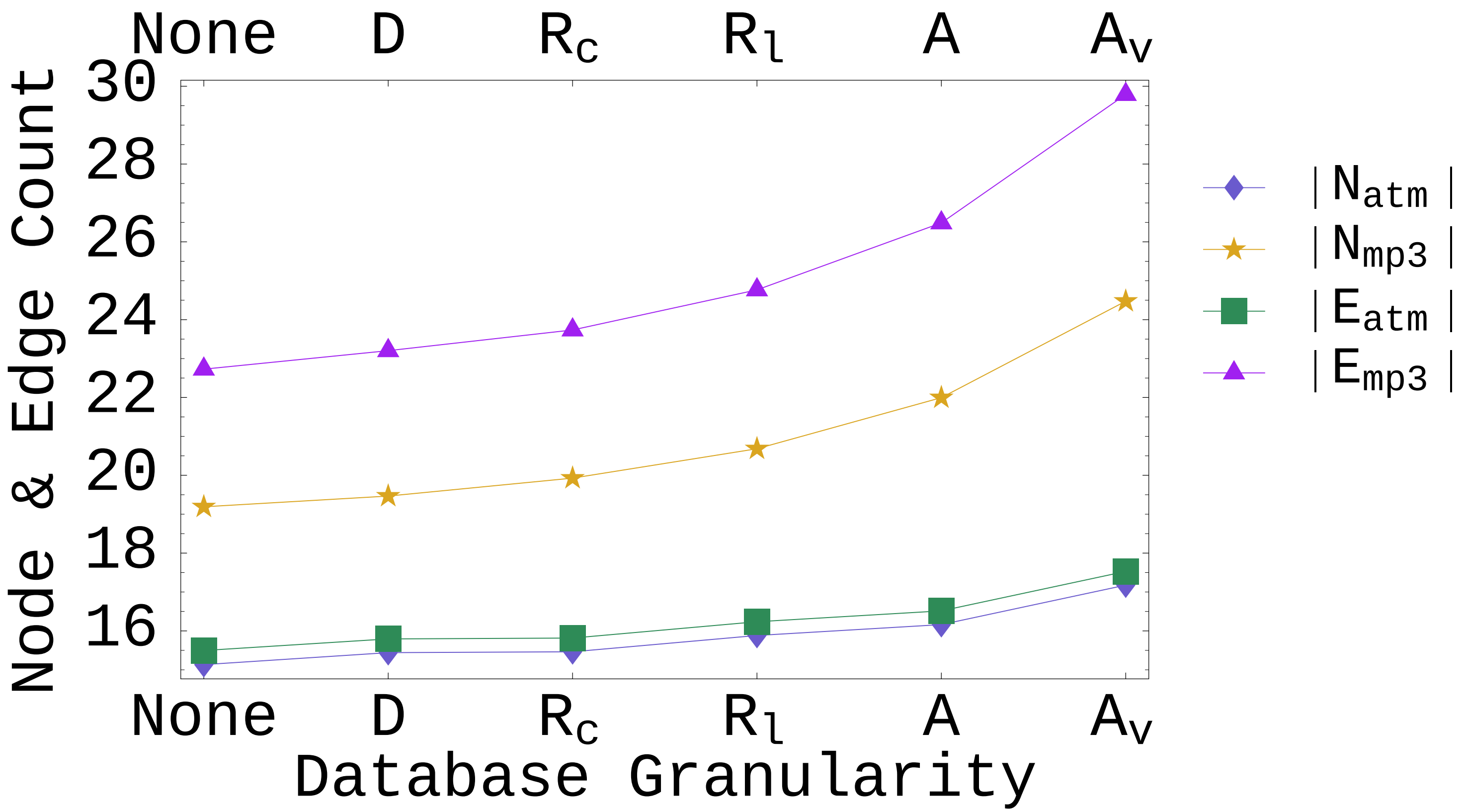
- DIAs at attribute value level represent 16.8% of `mp3cd`'s and 9.6% of ATM's total number of intraprocedural associations

Measuring Time Overhead



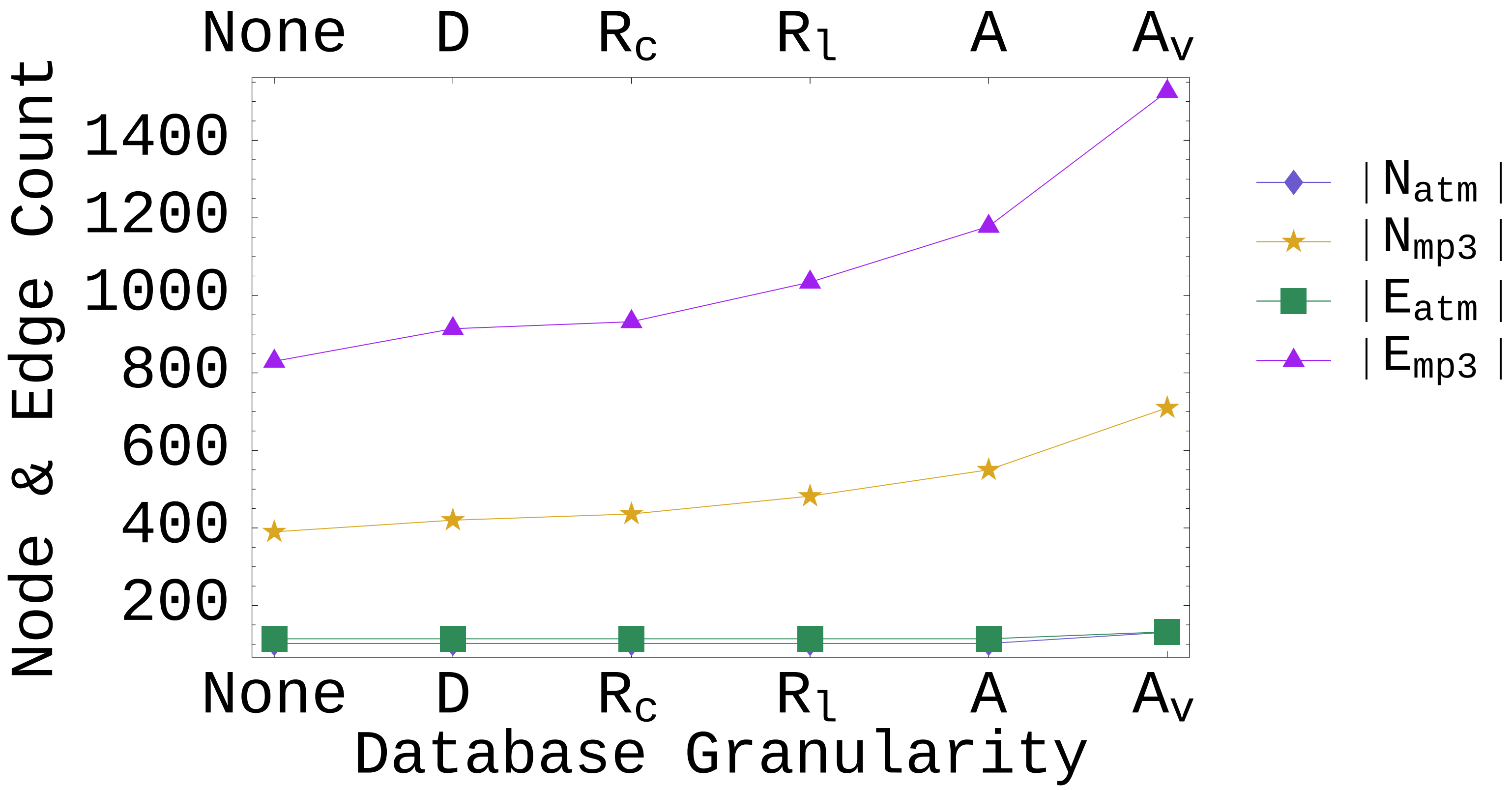
- Computing DIAs at the attribute value level incurs no more than a 5 second time overhead

Measuring Average Space Overhead



- mp3cd shows a more marked increase in the average number of nodes and edges than ATM

Measuring Maximum Space Overhead



- mp3cd shows a significantly greater maximum space overhead than ATM

Conclusions

- Must test the program's interaction with the database
- Unique family of test adequacy criteria to detect type (1) violations of database validity and completeness
- Intraprocedural database interactions can be computed from a DICFG with minimal time and space overhead
- High number of hanging definitions indicates that the scope of data flow analysis could be broadened
- This data flow-based test adequacy criteria can serve as the foundation for algorithms that measure test suite quality, automatically generate test cases, and support regression testing

`http://cs.allegheny.edu/~gkapfham/research/diatoms/`