from left to right:

Kaitlin McAbee
Carl Chapman
Katie Stolee*
Cody Hoover
Cole Groff
Trevor Lund

Satsy

May14
13

an example-based search engine for source code

Input

Output

Search

Problem Statement

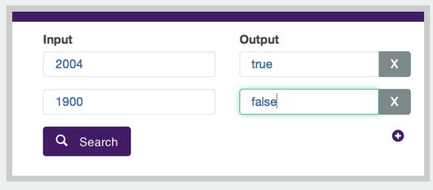
Satsy searches a database of Java methods based on their ability to satisfy a user-specified input-output pair (**IOPair**). Our client, Katie Stolee*, developed the prototype Satsy program as her PhD thesis project. Katie's prototype was single threaded and used a basic Java Swing user interface. She needed a more sophisticated user interface and concurrent execution. Our task was to build a web service based on her prototype that could utilize multiple cores and serve multiple users simultaneously. Katie is the primary user in the short term, as she will use our product to test improvements to the system. Programmers at large are the intended long-term users, who may benefit from a new way to search existing code by functionality.

Example Usage

Find code that determines if a year is a leap year

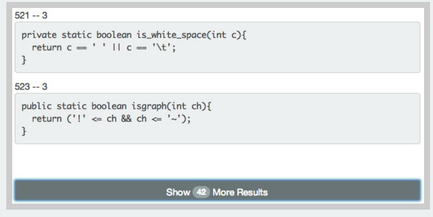
Step 1

Enter IOPairs, press 'Search'

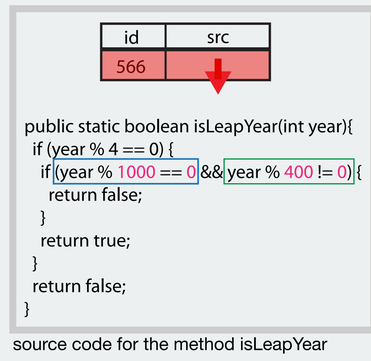


Step 2

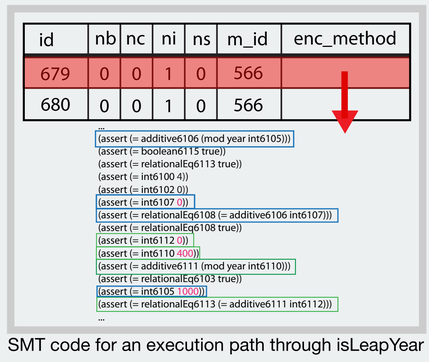
Press 'Show more Results' as needed



Encoding Execution Paths in SMT



source code for the method isLeapYear



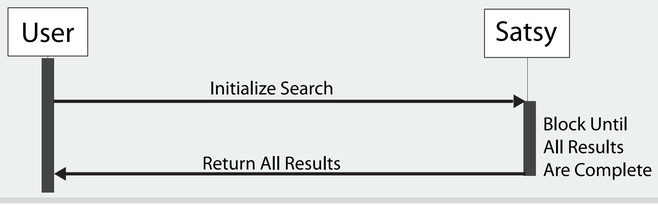
SMT code for an execution path through isLeapYear

The method 'isLeapYear' has two execution paths, stored as separate database entries. Each path is encoded in SMT as a set of first order logic expressions. After binding the IOPair to the encoded path, **Z3** can determine if the path will satisfy the IOPair specification.

Speed Limitations

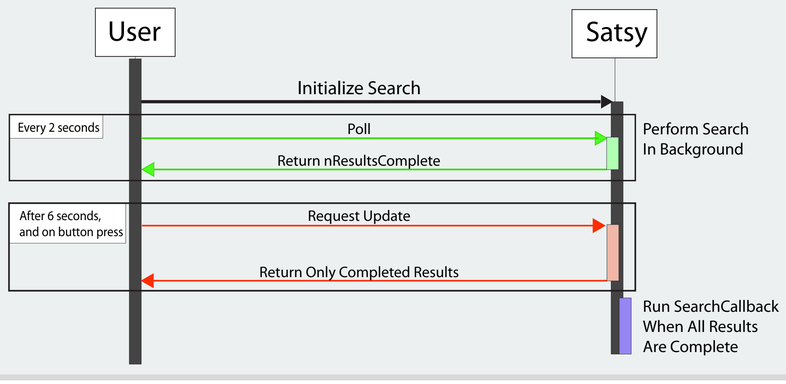
A search is composed of many solvers. Each solver has to bind an IOPair to an encoded path, execute Z3 on the resulting SMT file, and record a result. This always takes some minimum amount of CPU time - (about 60 milliseconds on a modern CPU). Searching through 1000 paths with 1 IOPair takes about 1 minute of CPU time.

Our original, naive implementation blocked until all paths had been searched and then returned all results at once.

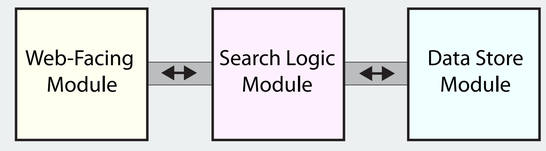


Designing Around Our Speed Limit

We want to provide our users with some results to look at as soon as possible, without limiting the size of the database. Our solution is to provide the user access to all completed results while the search continues.



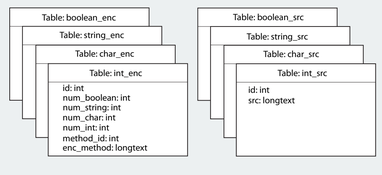
Modular Design



Satsy consists of three abstract modules, separating storage, logic and presentation concerns.

Database Design

Encoded paths and the source code for their methods are stored in separate groups of tables, which are partitioned by output type.



What is Z3?

Z3 is a theorem prover developed by Microsoft. Given a system of values, functions and constraints, Z3 can determine if the system is satisfiable or not.

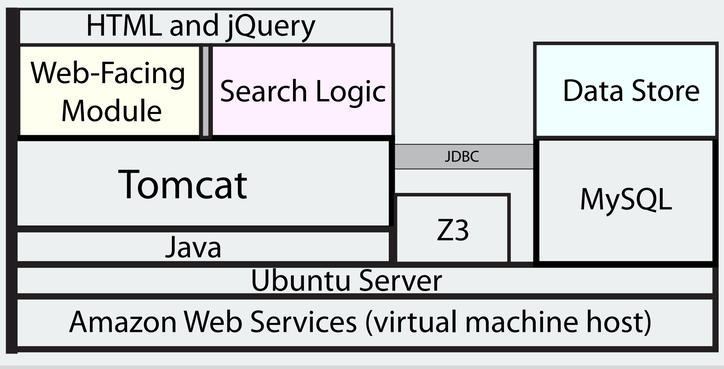
Non-Functional Requirements

- timeout per search
- timeout per solver
- threads per search
- trace level logging
- detail level logging

Testing Strategy

- JUnit testing for ranking and POJO classes
- Automated testing for validation and profiling
- Manual testing for GUI and solver development

System Portrait



Search Logic Module Functions

Initialize Search
bind each IOPair to each encoded path in a separate solver
prepare a collection of Result objects - one for each method
execute solvers concurrently, writing sat or unsat to a Result
once all solvers for a method are complete, that Result is complete

Poll
return count of completed Results

Request Update
copy complete Results into a list
rank and return the list

Ranking

When a user requests an update, all completed results are ranked and returned. We developed ranking algorithms based on the number of satisfied paths, percentage of satisfied paths, and whether all paths were satisfied or not. Developing more sophisticated ranking algorithms is an issue open to further research.