CodeHint

Paper by Joel Galenson, Philip Reames, Rastislav Bodik, Björn Hartmann, Koushik Sen

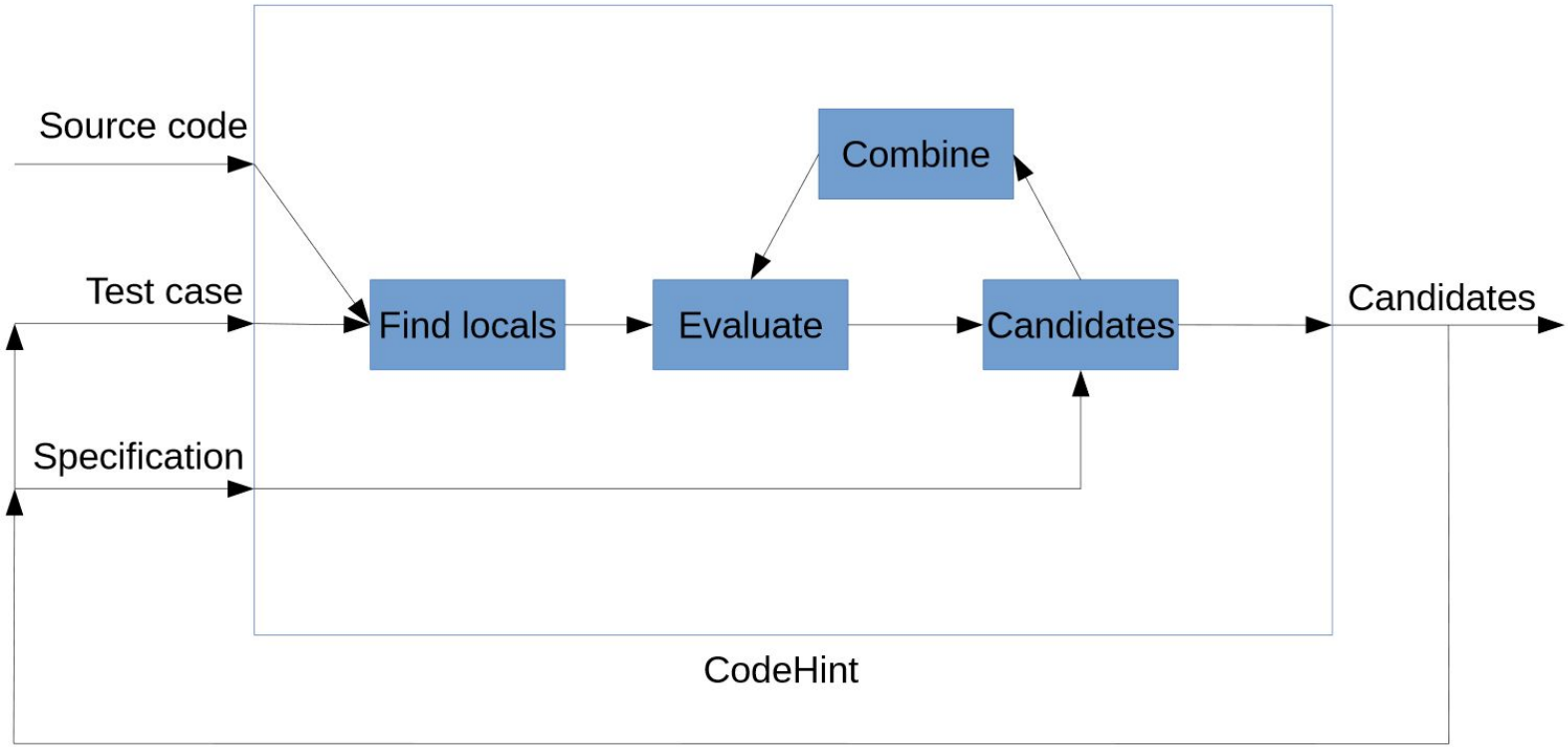
slide author names omitted for FERPA compliance

What is CodeHint?

CodeHint - Generates missing Java code. Actually finds and evaluates methods and statements (including I/O, reflection & other advanced features)

- Dynamic
- Easy to use
- Interactive

Overview



[Galeson, J., Future Programming Workshop, Strange Loop, 2014]

So how does CodeHint work?

- **pdspecs** - get partial information of the desired result
- **Skeletons** - narrows search space using “holes”
- **Probabilistic model** - common methods and fields used in real-world Java

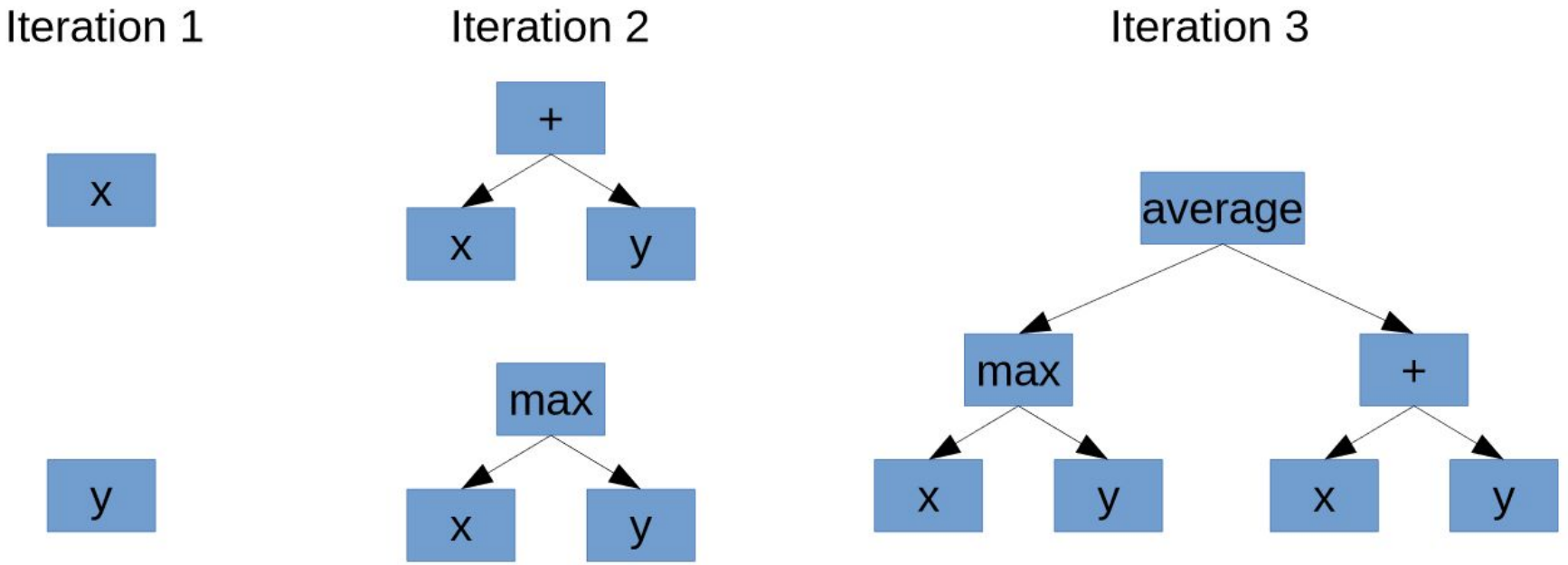
$$P(m|T)P(T) = \frac{\#\text{ accesses of } m \text{ on } T}{\#\text{ of accesses on } T} \times \frac{\#\text{ of accesses on } T}{\#\text{ of accesses}}.$$

Probability of accessing method *m* on type *T*

Simplistic approach.

Avoids unlikely expressions.

3-Stage Algorithm (3 iterations)



[Galeson, J., Future Programming Workshop, Strange Loop, 2014]

Refinement

- Once the program computes the 5 most probable code lines the user can:
  - **Choose one of the expressions** (at any of the iterations)
  - **Discard all the expressions** and test with other inputs
  - **Rerun the program** with different input to evaluate if some of the candidates fail and continue the iteratively with different inputs each time

User Perspective Scenario

- How does this apply in practice?
- Have a graphical tree of elements
- Want to find an element by clicking
- Do not know the API or what type the element is
- Hard to find answer using traditional means

Using CodeHint

- With CodeHint, can set breakpoint in appropriate place
- Click on element with name “Alice”
- Can use knowledge of Java’s toString method
- Enter pdspec o’.toString.contains(“Alice”)
- CodeHint now lists 8 expressions meeting this pdspec

Refining CodeHint Results

- To refine, look for “Bob” element
  - o’.toString().contains(“Bob”)
- Down to 7 results
- Most results have type TreePath
  - o’ instanceof TreePath
- Click outside of tree, no elements
  - o’ == null
- Only one option left
  - o = ((JTree)tree).getPathForLocation(x, y); which is correct

In Action - Traditional Method

In Action - CodeHint

CodeHint Skeletons

- Want data from clicked element
- getPathComponent of TreePath (o's type)
- Enter o.getPathComponent(??) skeleton
  - ?? represents missing code
- Set breakpoint, run CodeHint, click "Eve" element
- Enter pdspec _rv'.toString().contains("Eve")
  - -rv' is return value of expression
- CodeHint will return integers that fit the pdspec
  - o.getPathComponent(e.getClickCount())

Contributions

- A new method for synthesizing code that is dynamic, easy to use, and interactive.
- An efficient algorithm that uses the dynamic content to generate candidate statements.
- An implementation of this algorithm as an Eclipse plugin.
- Empirical evaluations and user studies.

Study 1

Study 1: Line-level Tasks

- Three scenarios, five sub-tasks each based on real coding issues.
  - Parse: manipulate strings
  - Swing: create a small GUI
  - RandomWriter: create a Markov model to generate random input text examples
- Only 13 tasks were solvable with CodeHint.

Conditions of Study 1

Each subject was assigned a condition:

- Control- Can’t use CodeHint
- Experimental- Must use CodeHint (unless it fails)
- Choice- User’s choice

Control and experimental were randomly assigned for first two scenarios, with Choice always being assigned last.

Study 2

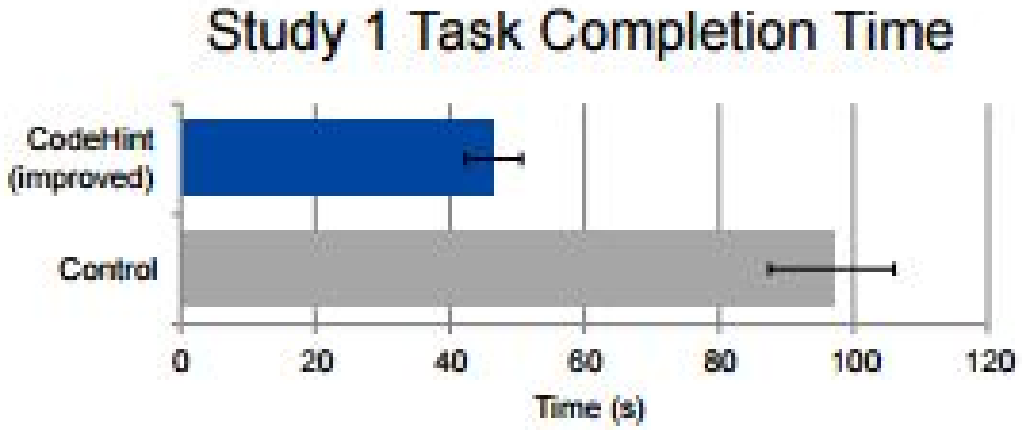
Study 2: Open-ended Tasks

- More difficult than the first. Tasks involved writing many lines of code with complex APIs. Twenty minute time limit per task.
- Featured two scenarios with three tasks each.
  - Eclipse: Implement a simple Eclipse profiler plugin.
  - Note: Write a GUI note-taking application.
- Only featured Control and Choice. Users with the Control condition were allowed to use CodeHint if necessary.

Participants

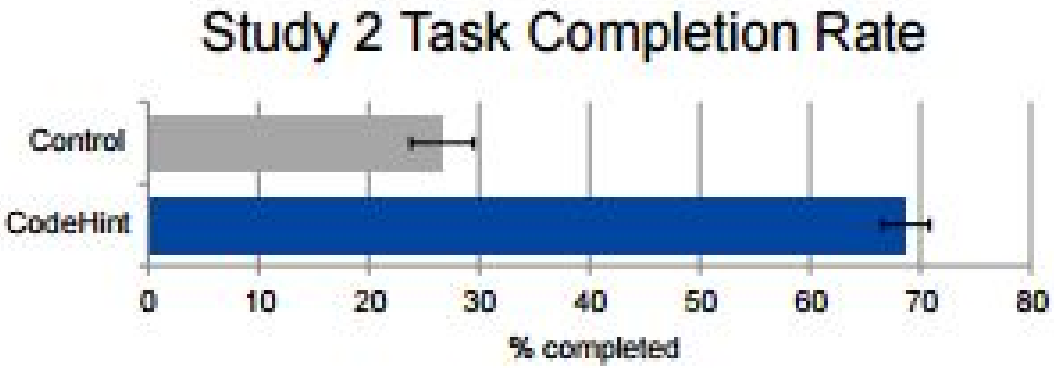
- Study 1 began with nine subjects, with five more brought in to complete only an updated version of Choice. Featured twelve grad students and two undergrads at UC Berkeley.
- Study 2 had fourteen subjects. Used an improved version of CodeHint. Featured ten undergrads and four grad students.
- None of the subjects in either study had used CodeHint before.

Completion Time



In all but one case, participants in the first study completed all tasks. Subjects using CodeHint completed tasks more than twice as fast.

Completion Rate



Participants in Study 2 did not complete many tasks because of the increased difficulty. Subjects using CodeHint were more than twice as successful as those not using it.

Quality and Choice

- In Study 1, CodeHint = 11 bugs in 122 tasks solved.
  - No CodeHint = 24 bugs in 93 tasks solved.
- In the Choice condition, participants chose to use it 71% of the time.
- In a post study questionnaire, subjects rated the overall usefulness of CodeHint an average of 7.7 out of 10.
  - Six subjects even asked for the plugin after completing their tasks.

Pdspecs in Both Studies

- CodeHint presented a small number of candidates after the initial pdspec
  - Study 1: Average = 13, Median = 2
  - Study 2: Average = 34, Median = 3
- When refining candidates, pdspecs reduced candidate list by 31% on average
  - 66% if lists of equivalent candidates are not counted
- The pdspecs were also very successful in finding correct candidates
  - Study 1: 53% had desired value, 33% had desired type, 14% arbitrary predicates
  - Study 2: 51% had desired value, 38% had desired type, 10% arbitrary predicates

Discussion Questions

Does CodeHint promote lazy programmers, who do not need to learn the APIs and language?

Discussion Questions

Does CodeHint promote lazy programmers, who do not need to learn the APIs and language?

> It can be said that users who use CodeHint learn the proper way to accomplish tasks, and do so faster. This will help develop them as programmers until they no longer need CodeHint.

Discussion Questions

Would developers use CodeHint over the ways that they are used to, and that work for them?

Discussion Questions

Would developers use CodeHint over the ways that they are used to, and that work for them?

71% of participants found CodeHint useful, and all participants expressed interest in using CodeHint for their own development.

Discussion Questions

Would developers trust CodeHint to solve tasks they find difficult?

Discussion Questions

Would developers trust CodeHint to solve tasks they find difficult?

Study of real world Java applications and Stack Overflow code shows that 99% of Java statements are within 4 depths (x is depth 1, foo.bar(x,y) is depth 2, and so on), and thus within the range findability by CodeHint.

Discussion Questions

CodeHint works for Java, but will it work with other languages?

Discussion Questions

CodeHint works for Java, but will it work with other languages?

CodeHint works well with dynamic languages, and support for other languages intended in the future. Javascript version of CodeHint completed and launched after the publication of the paper [Galeson, J., CodeHint].

Discussion Questions

The two studies were only done with a combined 28 people, is this enough real-world testing to describe CodeHint as beneficial?

Discussion Questions

The two studies were only done with a combined 28 people, is this enough real-world testing to describe CodeHint as beneficial?

> As mentioned above, code from real world Java applications (amounting to over ten million lines of code), and 3,333 lines of Stack Overflow code were analyzed and found to be within the level of sophistication findable by CodeHint.