

# A Canonicalizing Model for Building Programming Tutors

Kelly Rivers and Kenneth R. Koedinger

# Time Constraints in Authoring

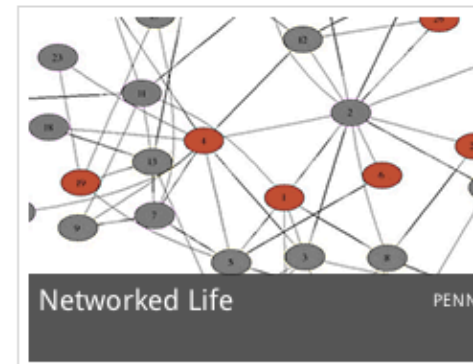
## Healthcare, Medicine, and Biology

[see more](#)



## Society, Networks, and Information

[see more](#)



## Economics, Finance, and Business

[see more](#)





# Proposed Fix: Large Sets of Data

---





# A Complex Problem

---





# Solution

---

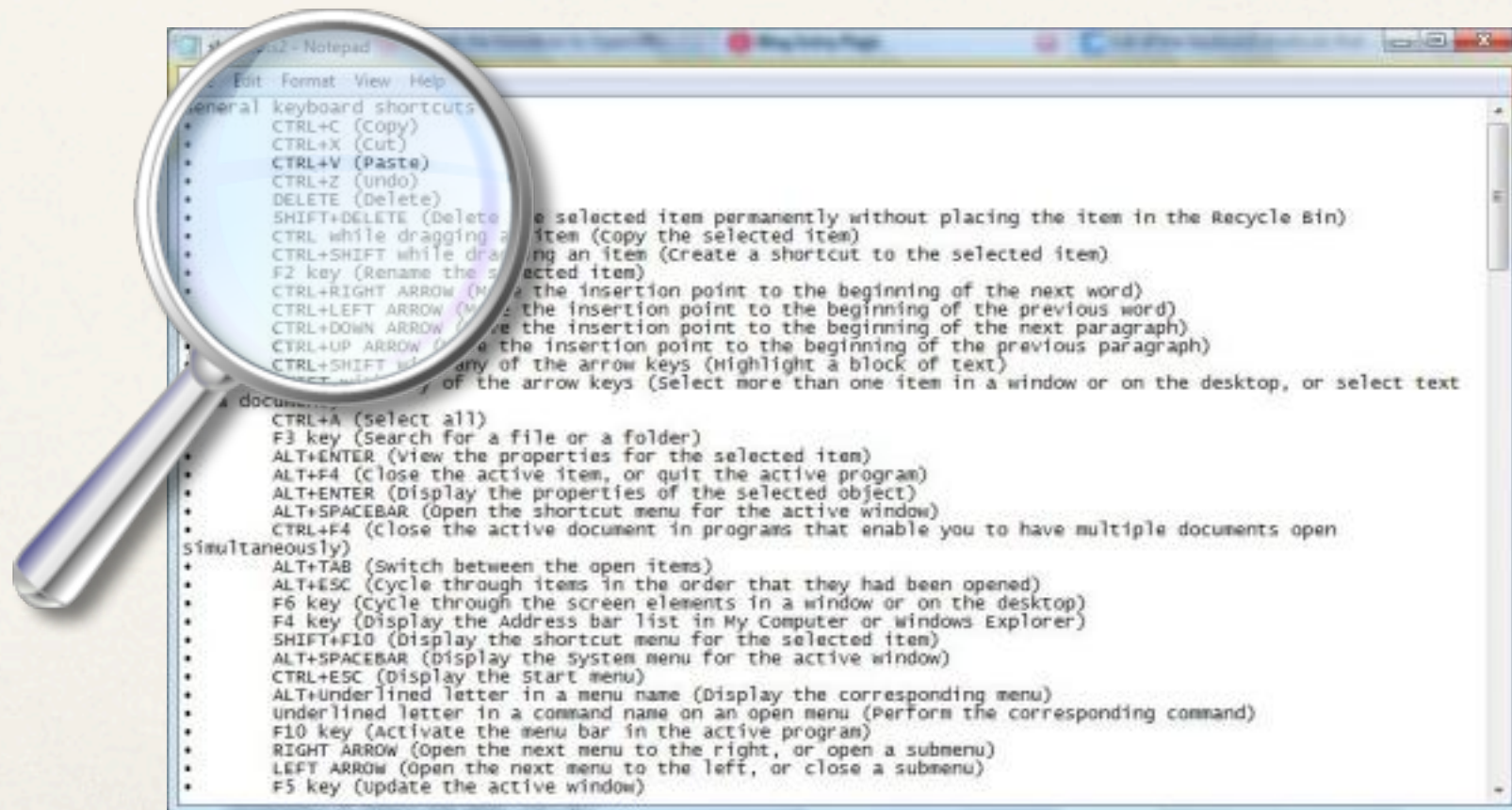
Instead of making more complex tutors, make the student submissions more condensed!



# Canonicalization

---

- ❖ A process for converting data that has more than one possible representation into a "standard", "normal", or canonical form

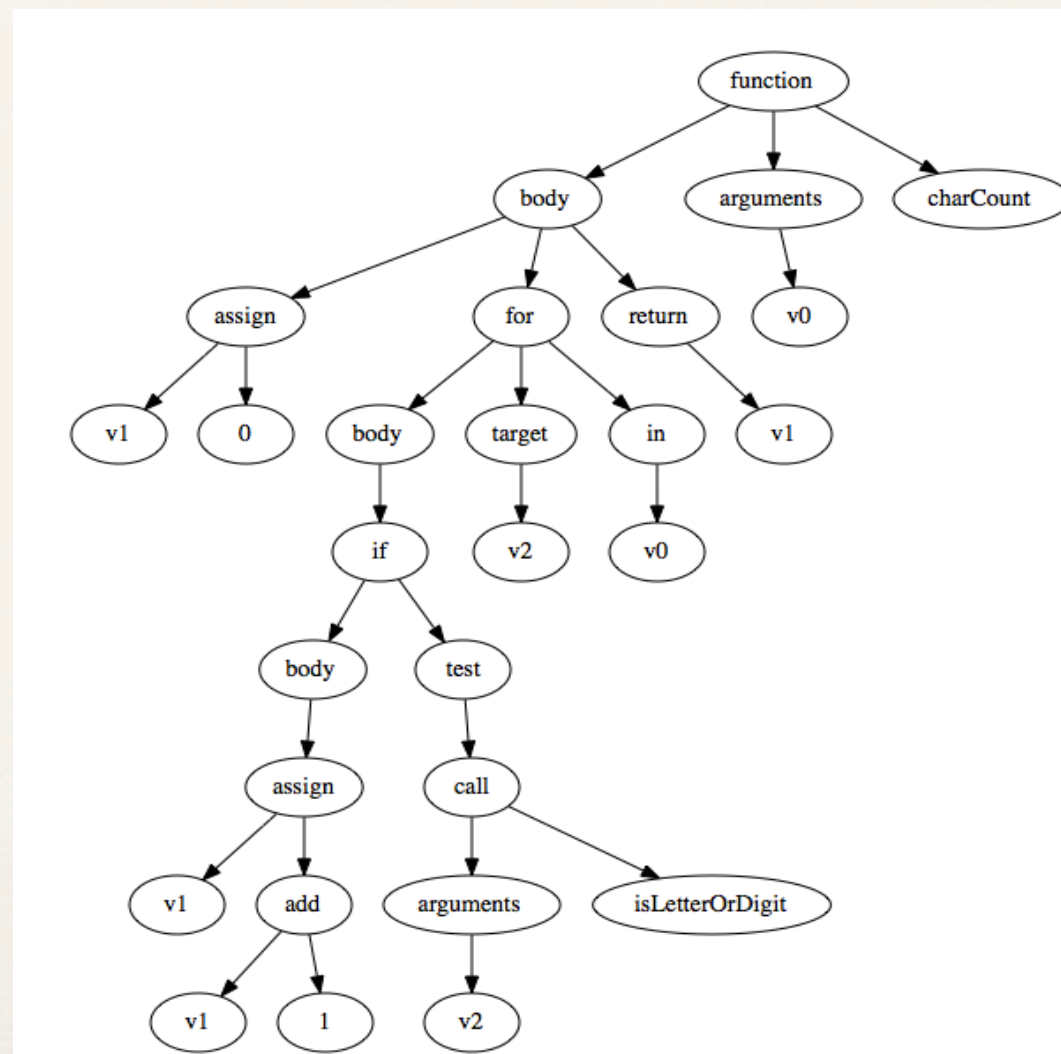




# Abstract Syntax Trees (ASTs)

---

- ❖ A tree representation of the abstract syntactic structure of source code written in a programming language



# Program Transformations

---

- ❖ Functions which are run on computer programs to change their content
- ❖ These usually guarantee that the result will be *semantically* equivalent



# The Canonicalization Process

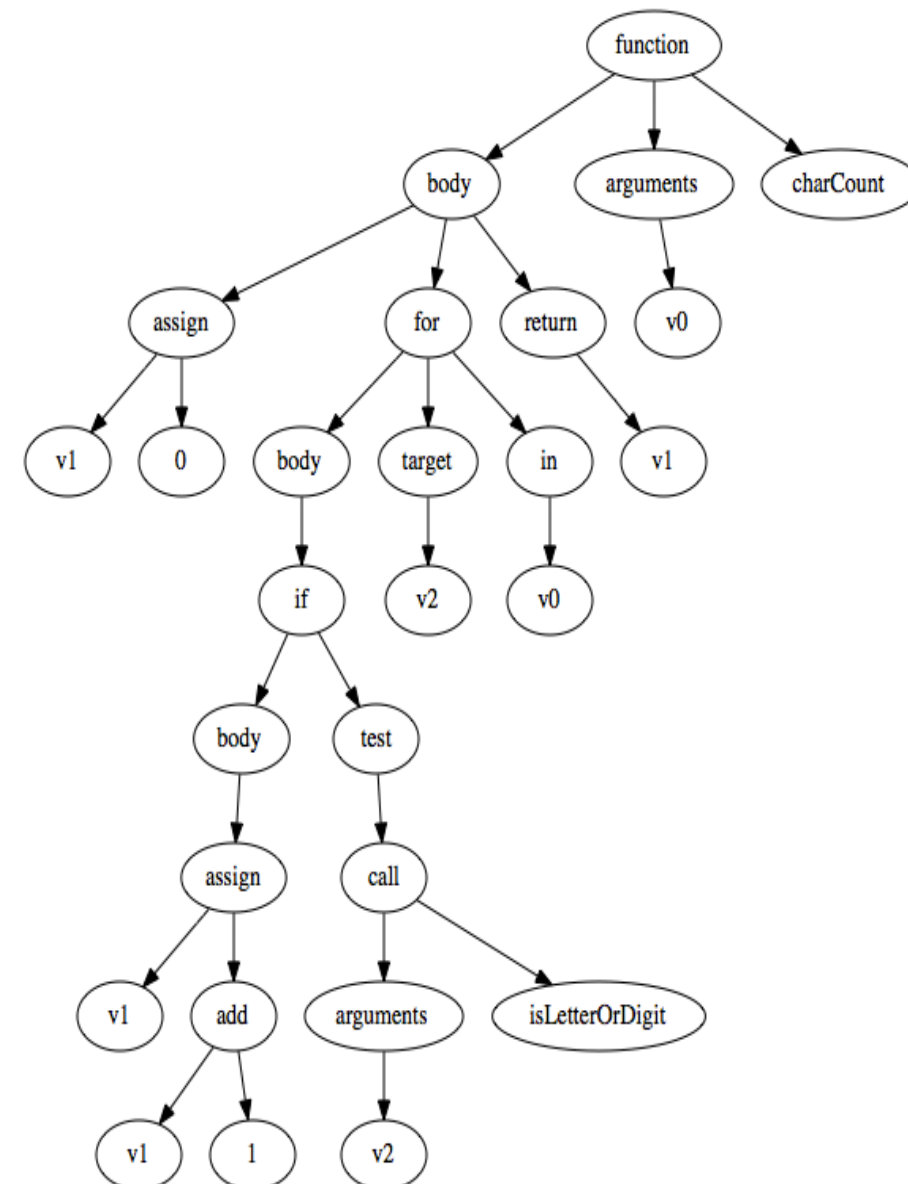
---



# Step 1: Student Code -> AST

```
def charCount(text):  
    count= 0  
    for c in text:  
        if isLetterOrDigit(c):  
            count +=1  
    return count
```

```
def charCount(text):  
    charcount = 0  
    for char in text:  
        if isLetterOrDigit(char): charcount += 1  
    return charcount
```





# Step 1: Student Code -> AST

---

```
def charCount(text):  
    count= 0  
    for c in text:  
        if isLetterOrDigit(c):  
            count +=1  
    return count
```

---

```
def charCount(text):  
    charcount = 0  
    for char in text:  
        if isLetterOrDigit(char): charcount += 1  
    return charcount
```

```
def charCount(v0):  
    v1 = 0  
    for v2 in v0:  
        if isLetterOrDigit(v2):  
            v1 += 1  
    return v1
```





# Step 2: Run Canonicalization Suite on AST

---

```
def charCount(v0):  
    v1 = 0  
    for v2 in v0:  
        v3 = isLetterOrDigit(v2)  
        if (v3 == True):  
            v1 = (v1 + 1)  
    return v1
```



```
def charCount(v0):  
    v1 = 0  
    for v2 in str(v0):  
        if (isLetterOrDigit(v2) == True):  
            v1 += 1  
    return v1
```

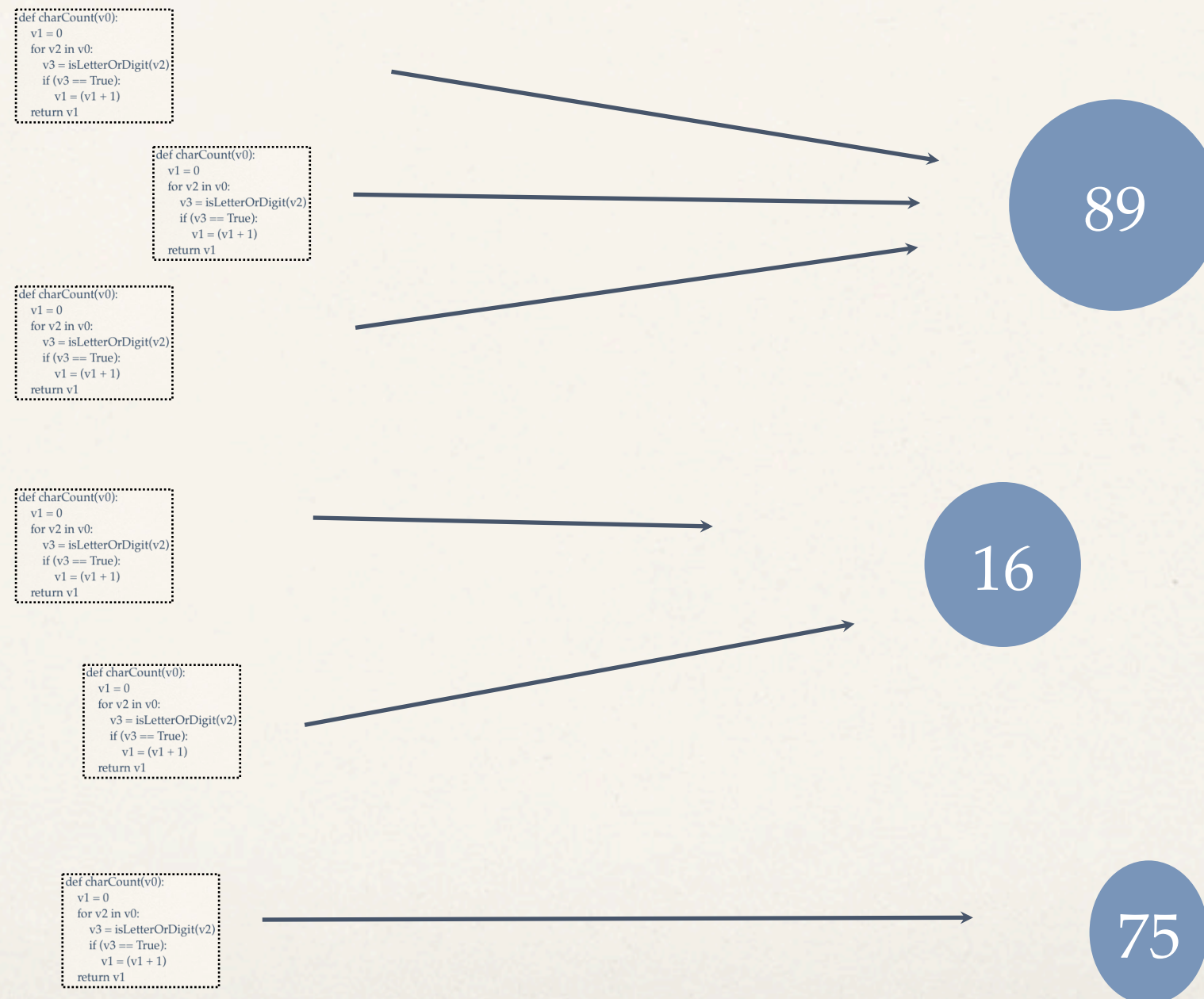


```
def charCount(v0):  
    v1 = 0  
    for v2 in v0:  
        if isLetterOrDigit(v2):  
            v1 = (1 + v1)  
    return v1
```



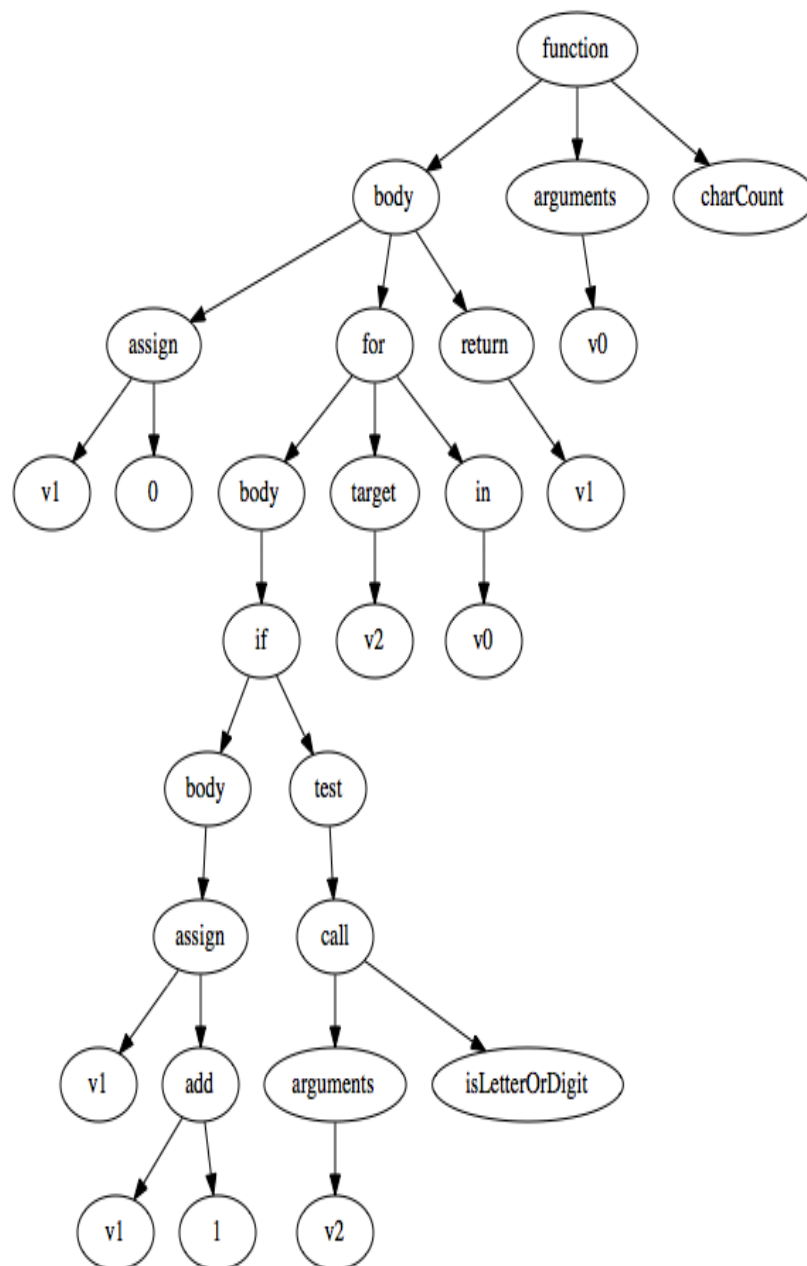


# Step 3: ASTs -> Canonicalized Models





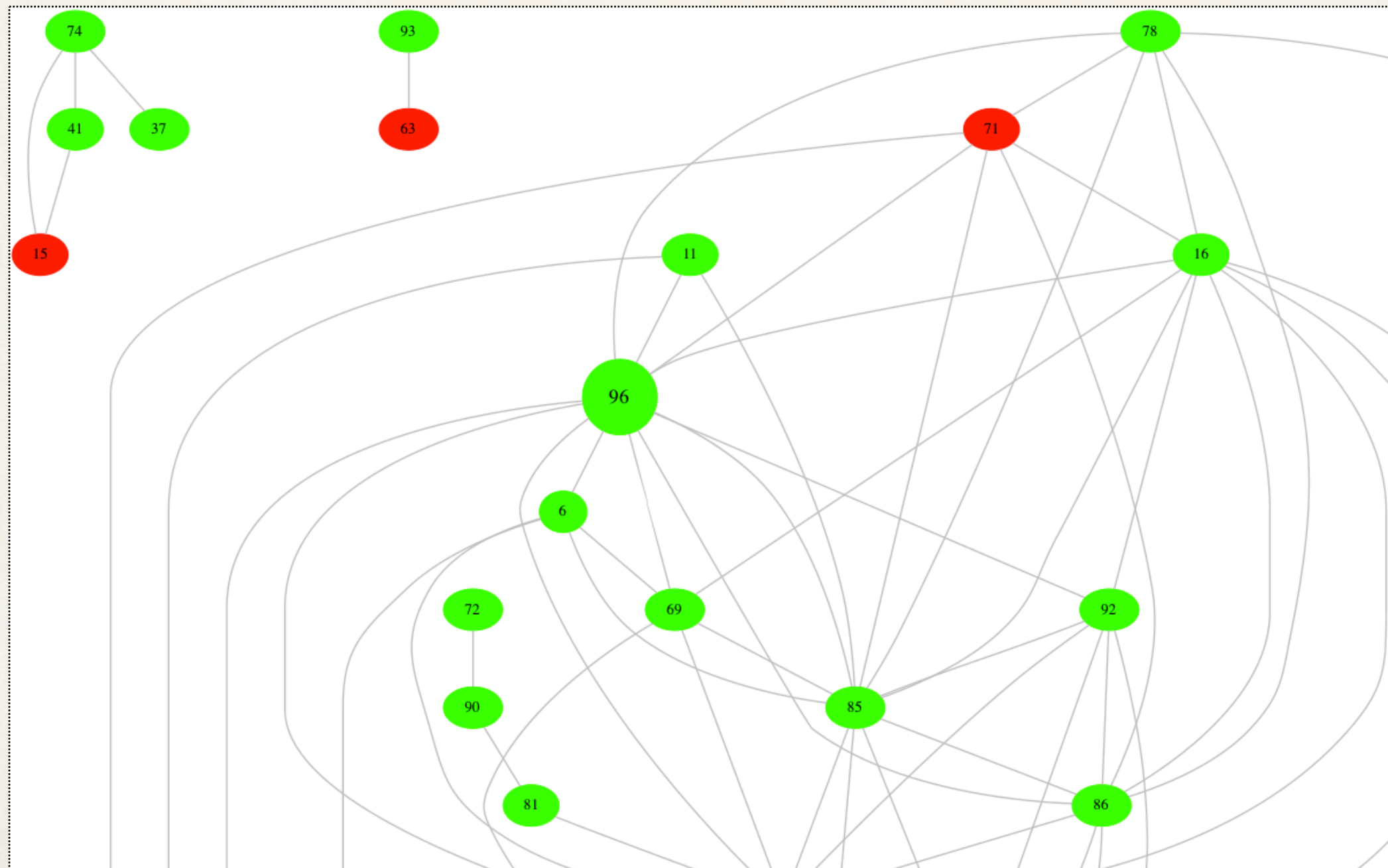
# Step 4: Canonicalized Models -> Source Code



```
def charCount(v0):
    v1 = 0
    for v2 in v0:
        if isLetterOrDigit(v2):
            v1 = 1
    return v1
```



# Step 5: Cluster Source Code



Context - **Approach** - Findings - Hint Creation - Conclusion



# Preliminary Results

---

Observe changes from Program Text -> AST -> Canonicalized Models



# Data Set

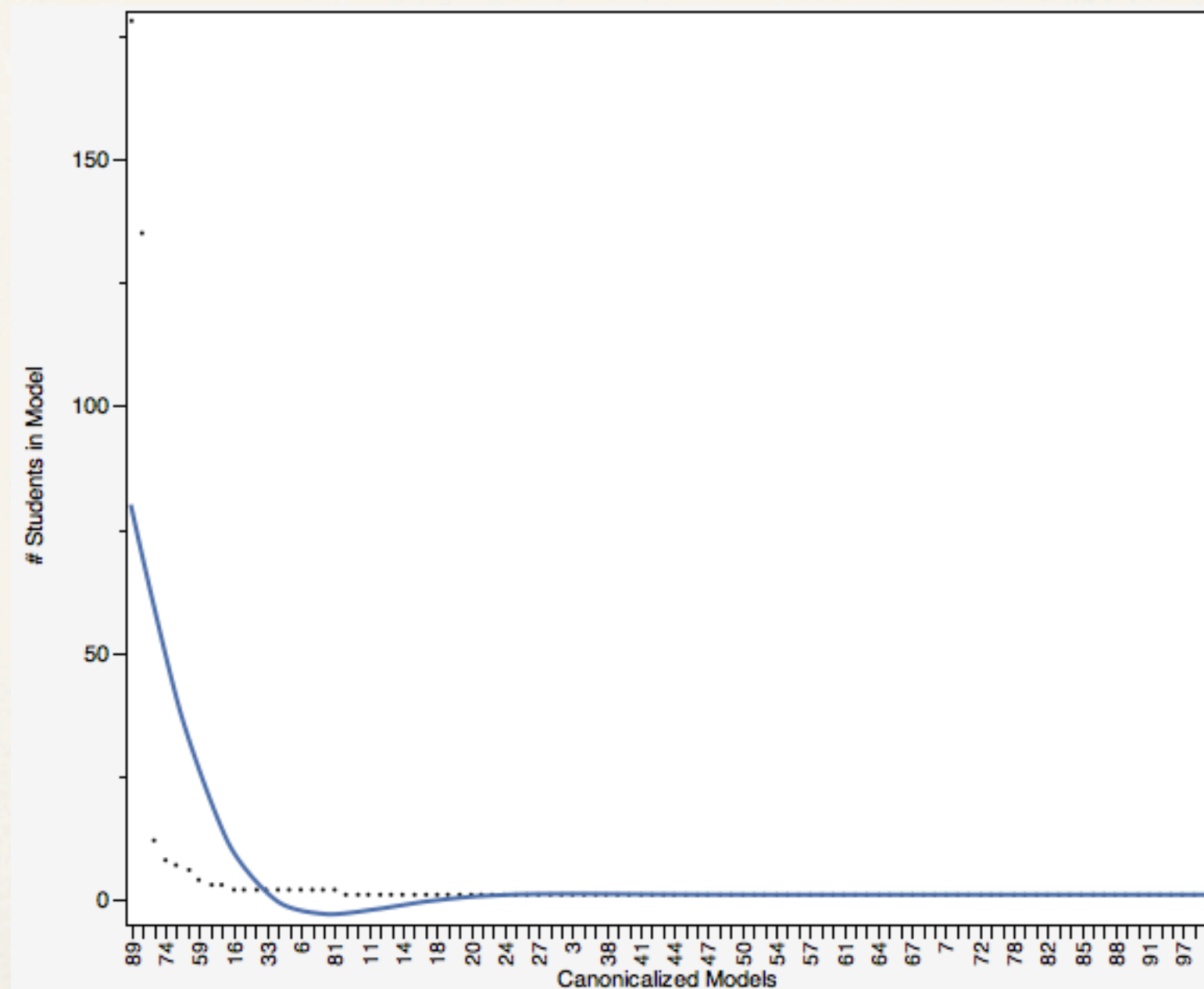
---

- ❖ Principles of Computing, Carnegie Mellon University, Spring 2011
- ❖ Language: Python (dynamic typing, multiple paradigms)
- ❖ Programming Assignments 1-3 (out of 6)
- ❖ 33 problems (14 solo, 17 collaborative, 2 bonus)
- ❖ Average of 22% incorrect submissions



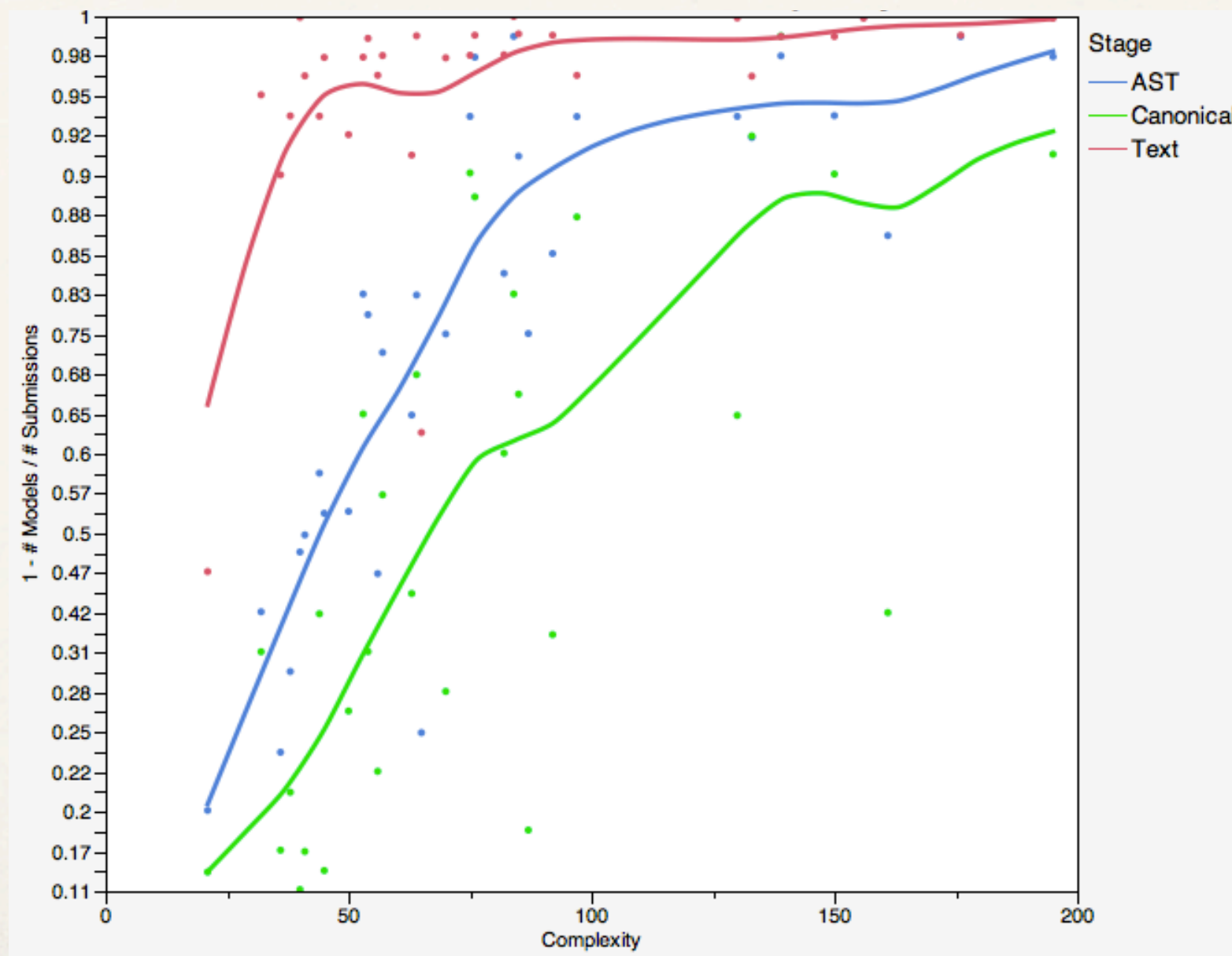
# The Long Tail

---





# Pre-Post Canonicalization





# Overall Statistics

---

	%Reduction	$\Delta$ Max Group Size	# Groups / # Individual
Text	5%*	-	9 / 387
AST	28%**	26**	29 / 263
Canonical Models	48%**	94**	26 / 176

T-test of means, \* < 0.01, \*\* < 0.001



# Future Work

---

Use the simplified, canonicalized models to create tutors per problem



# Traditional Hint Construction

hw2, Problem simplePigLatin

Assignment: hw2 Problem: simplePigLatin

Save Grades Create Gradesheet

Percentage Similarity: 0.9

Students in Canonical Form: 9

35	1-1-1-1	2	4
36	1-1-1-1	1	4
37	1-1-1-1	1	4
38	1-1-1-1	1	4
39	1-1-1-1	1	4
40	1-1-1-1	1	4
41	1-1-1-1	21	4
42	1-1-1-1	1	4
43	1-1-1-1	1	4
44	1-1-1-1	1	4
45	1-1-1-1	1	4
46	1-1-1-1	1	4
47	1-1-1-1	2	4
48	1-1-1-1	24	4
49	0-0-0-0	1	0
50	1-1-1-1	1	4
51	1-1-1-1	1	4
52	1-1-1-1	1	4
53	1-1-1-1	1	4
54	1-1-1-1	212	4
55	1-1-1-1	1	4
56	1-1-1-1	1	4
57	1-1-1-1	84	4
58	1-1-1-1	1	4
59	1-1-1-1	1	4
60	1-1-1-1	1	4
61	1-1-1-1	1	4
62	1-1-1-1	1	4
63	0-0-0-0	1	0
64	1-1-1-1	11	4
65	0-0-0-0	1	0
66	1-1-1-1	1	4
67	1-1-1-1	1	4

Canonical Form: 57

def simplePigLatin(v0):  
 return (v0[:v0.find(' ')] + (v0[v0.find(' ')] + 'ay ' +

Student's Code

def simplePigLatin(v0):  
 v1 = v0.find(' ')  
 v2 = v0[0:v1]  
 v3 = v0[v1 + 1:len(v0)]  
 v4 = ((v2[1:len(v2)] + v2[0]) + 'ay')  
 v5 = ((v3[1:len(v3)] + v3[0]) + 'ay')  
 return ((v4 + ' ') + v5)

simplePigLatin Visualization

Context - Approach - Findings - Hint Creation - Conclusion



# Automatic Hint Construction: Model Based

---

When using splicing, default terms can be successfully left blank!

```
def simplePigLatin(v0):  
    v1 = v0.find(' ')  
    v2 = v0[0:v1]  
    v3 = v0[v1::None]  
    v4 = (v2[0] + 'ay')  
    v5 = (v3[1] + 'ay')  
    v6 = (((v2[1::None] + v4) + ' ') + v3[2::None]) + v5)  
    return v6
```

```
def simplePigLatin(v0):  
    v1 = v0.find(' ')  
    v2 = v0[:v1]  
    v3 = v0[v1:]  
    v4 = (v2[0] + 'ay')  
    v5 = (v3[1] + 'ay')  
    v6 = (((v2[1:] + v4) + ' ') + v3[2:]) + v5)  
    return v6
```

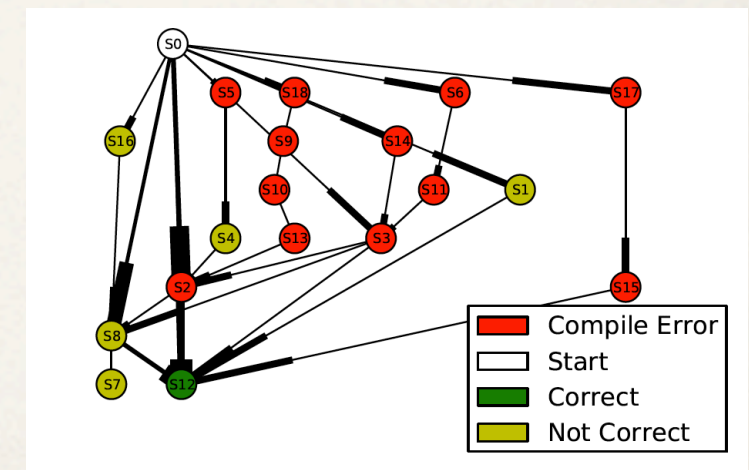


# Automatic Hint Construction: Data-Driven

- ❖ “Program Representation for Automatic Hint Generation for a Data-Driven Novice Programming Tutor” (Jin et al, ITS 2012)
- ❖ “Calculating Probabilistic Distance to Solution in a Complex Problem Solving Domain” (Sudol-DeLyser, Rivers, and Harris, EDM 2012)

```
def charCount(v0):  
    v1 = 0  
    for v2 in range(len(v0)):  
        if isLetterOrDigit(v0[v2]):  
            v1 = (1 + v1)  
    return v1
```

```
def charCount(v0):  
    v1 = 0  
    for v2 in range((len(v0) - 1)):  
        if isLetterOrDigit(v0[v2]):  
            v1 = (1 + v1)  
    return v1
```





# Automatic Hint Construction: Crowdsourced

reddit PROGRAMMING **comments** related other discussions (1) view images(0)

137 submitted 21 hours ago by OvidPerl (blogs.perl.org)

63 comments share source save hide report hide all child comments

all 63 comments [subscribe](#)

sorted by: **best**

navigate by: [submitter](#) | [moderator](#) | [friend](#) | [admin](#) | [IAmA](#) | [images](#) | [popular](#) | [new](#)

Commenting as: kcreeks

**Bold** *Italic* ~~strike~~ <sup>sup</sup> [Link](#) [Quote](#) [Code](#) • [Bullets](#) [1.Numbers](#) [reddiquette](#) [macros](#)

[save](#) [formatting help](#)

[\[-\]](#) **teek** 17 points 16 hours ago (18|1)

Looks like there are some errors...

In Chapter 1:

Some people just do the following:

```
#!/perl
```

This does not work.

[permalink](#) [report](#) [source](#) [save](#) [reply](#) [hide child comments](#)

[\[-\]](#) **FoeHammer99099** 3 points 14 hours ago (3|0)

These are apparently the unedited chapters. If no one else has, it might be helpful for you to leave a comment and inform them of their mistake.

[permalink](#) [parent](#) [source](#) [report](#) [save](#) [reply](#)

[\[-\]](#) **HoWheelsWork** 2 points 10 hours ago (2|0)

FWIW, it does work on Windows, since it doesn't check the shebang for the perl path.

search reddit

this post was submitted on 14 Jun 2012

**137 points** (74% like it)

206 up votes 69 down votes

shortlink: <http://redd.it/v1ild>

**programming**

☒ Use subreddit style

[unsubscribe](#) [+shortcut](#) [+dashboard](#)

372,585 readers

☒ Show my flair on this reddit. It looks like:

kcreeks

[/r/programming](#) is a reddit for discussion and news about [computer programming](#)

**Guidelines**

- Please try to keep submissions on topic and of high quality.
- Just because it has a computer in it doesn't make it programming.
- Memes and image macros are not acceptable forms of content.
- If there is no code in your link, it probably doesn't belong here.
- App demos should include code and/or architecture discussion.
- Please follow proper [reddiquette](#).

**Info**

- Do you have a question? Check out [/r/learnprogramming](#), [/r/careerquestions](#) or [stackoverflow](#)



# Conclusion

---

- ❖ **What was done:** Simplifying input in complex domains with canonicalization
- ❖ **What comes next:** Automatic hint generation (through various techniques)
- ❖ **A question for you:** Can canonicalization be extended to further complex domains?



# Acknowledgements and Questions

---

**PIER**@ Carnegie Mellon  
PROGRAM IN INTERDISCIPLINARY EDUCATION RESEARCH



- ❖ Ken Koedinger

- ❖ For always providing excellent advice

- ❖ David Kosbie and Leigh Ann Sudol-DeLyser

- ❖ For their never-ending help with developing new ideas

- ❖ Gregory Dyke

- ❖ For drawing the picture of a tree

- ❖ This work was supported in part by Graduate Training Grant awarded to Carnegie Mellon University by the Department of Education (# R305B090023).



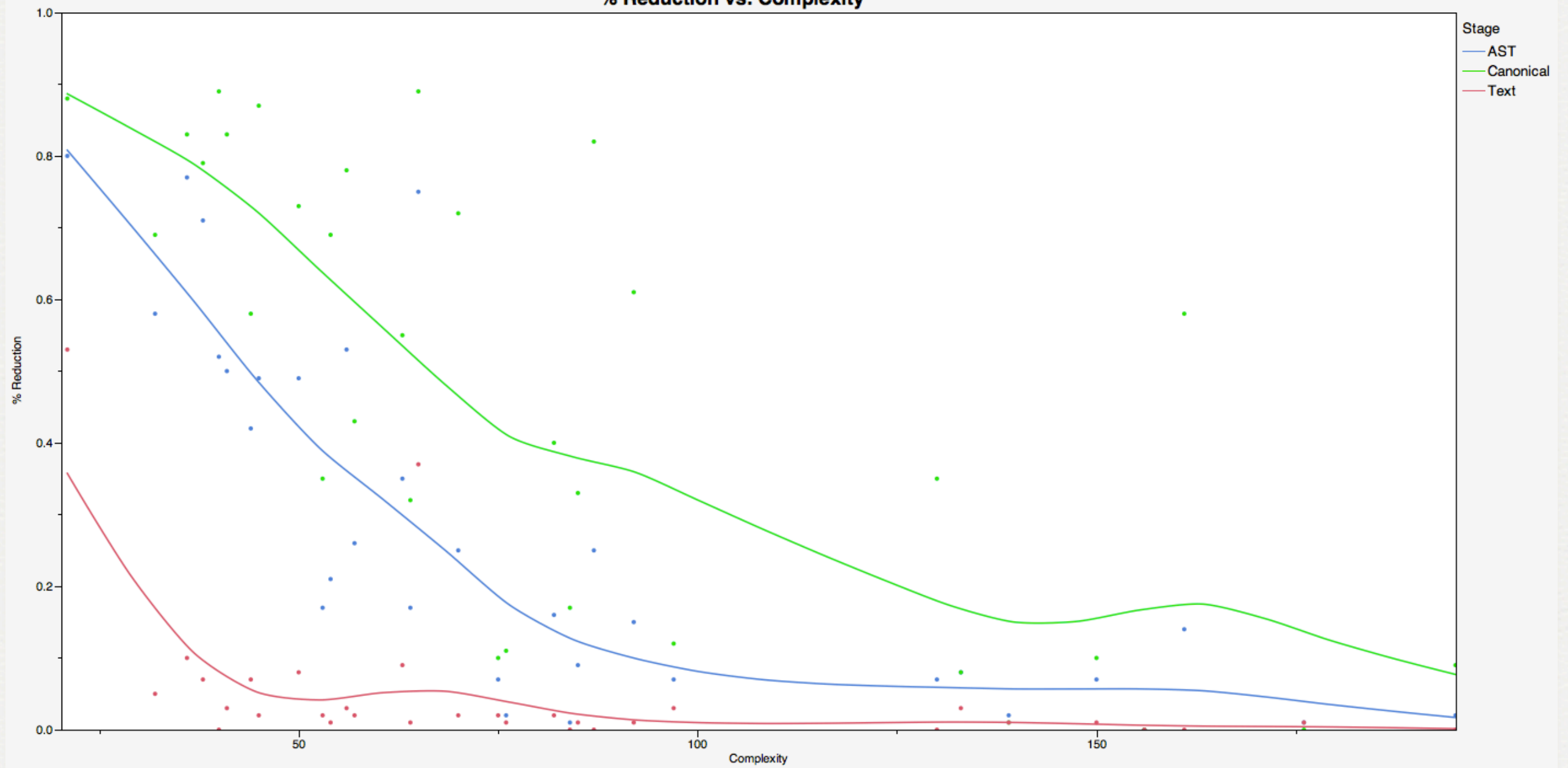
# Canonicalization Suite

---

```
functions = [lambda x: helperFolding(x, tree, problems),
             lambda x: simplify(x, argTypes, {}, [0]),
             constantFolding,
             lambda x : copyPropagation(x, tree, problems, {}),
             lambda x: collapseConditionals(x, tree, problems),
             constantFolding,
             lambda x: deadCodeRemoval(x, {}, tree, problems),
             lambda x : crashableCopyPropagation(x, [], tree, problems),
             deMorganize, removeUnneccessaryEquals, conditionalRedundancy,
             combineConditionals,
             lambda x: collapseConditionals(x, tree, problems),
             orderCommutativeOperations,
             constantFolding,
             cleanupBoolOps, cleanupRanges, cleanupSlices, cleanupTypes,
             cleanupInverts, cleanupNegations,
             deMorganize, orderCommutativeOperations
            ]
```



% Reduction vs. Complexity





# We are not breaking Rice's Theorem

---

