acm's women in computing
acm-w
ACM-W CIS Newsletter:
Celebrating, Informing & Supporting Women in Computing

In this issue

Research Spotlights

In this special issue of Research Spotlights, we highlight some of the Grace Hopper Celebration of Women in Computing 2014 ACM Student Research Competition participants and awardees.

"Eve eat dust mop": Data-driven measure of Child Language Development with Simple Syntactic Templates

By Shannon Lubetich
shannonlubetich@gmail.com

When assessing child language development, researchers have traditionally had to choose between easily computable metrics focused on superficial aspects of language, and more expressive metrics that are carefully designed to cover specific syntactic structures but require substantial and tedious labor. Recent work has shown that existing expressive metrics for child language development can be automated and produce accurate results. We go a step further and propose that measurement of syntactic development can be performed automatically in a completely data-driven way without the need for defining language-specific inventories of grammatical structures. As a crucial step in that direction, we show that four simple feature templates are as expressive of language development as a carefully crafted standard inventory of grammatical structures that is commonly used and has been validated empirically.

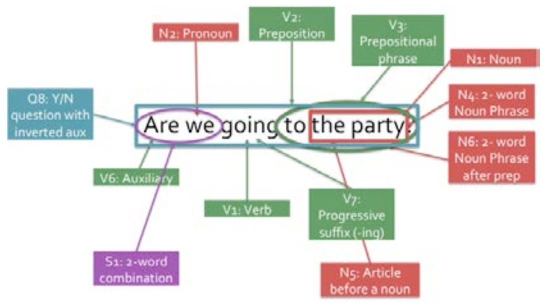
N2: Pronoun
V2: Preposition
V3: Prepositional phrase
N1: Noun
Q8: Y/N question with inverted aux
N4: 2-word Noun Phrase
Are we going to the party?
N6: 2-word Noun Phrase after prep
V6: Auxiliary
V1: Verb
V7: Progressive suffix (-ing)
S1: 2-word combination
N5: Article before a noun

Figure 1 A sentence depicting just 12 of 60 grammatical structures defined in a previous metric.

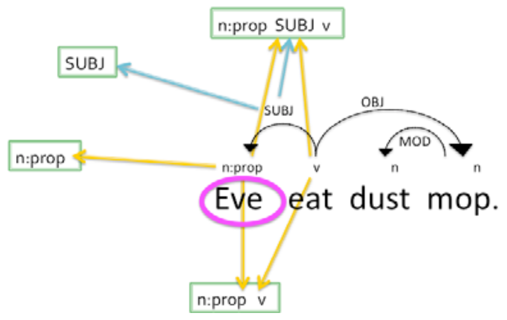

Figure 2 The four feature templates used in our data-driven approach.

More Information

The paper can be found here: http://aclanthology.info/papers/data-driven-measurement-of-child-language-development-with-simple-syntactic-templates.

About the Author

Shannon is a senior at Pomona College studying Computer Science and Linguistics. After graduation, she will join Google as a Software Engineer. She is motivated by the desire to solve interesting problems, specifically involving her loves of computers and languages. Shannon is interested in using computers to understand and generate natural language in order to advance human-computer interaction, and hopefully improve people's life experiences in our technological age. In her free time, she competes in scavenger hunts at Disneyland and DJs at her college's radio station.
Email: shannonlubetich@gmail.com
LinkedIn: www.linkedin.com/in/shannonlubetich
Twitter: @snl017 Instagram: @shannanogram

Using Escape Analysis to Speed Up Dynamic Race Detection

Emma Harrington
eh3@williams.edu

Multiple threads allow programs to increase performance by harnessing multiple cores but at the cost of greater programming complexity. Subtle bugs can manifest when multiple threads share resources simultaneously without proper synchronization. A race condition is one such bug and occurs when there are two concurrent accesses to a memory location and at least one of them is a write. Fortunately, programmers can use dynamic race detectors to alert them when data races occur. These detectors, however, degrade performance because they must track every access to every variable and verify those accesses are race free.

Many of these checks are unnecessary because a lot of objects are only accessed by a single thread and are consequently race-free. Escape analyses can identify these objects by tracking "thread-local" objects that are never accessible outside their creating thread. I implement a dynamic escape analysis to precisely eliminate all checks on thread-local objects. The figures below illustrate how the dynamic escape analysis identifies data that must be tracked by a precise race detector. I then develop a combined static and dynamic escape analysis that continues to identify all thread-local accesses while saving some run-time overhead by statically identifying thread-local objects whenever possible.

I found that about half of the objects are thread-local and 40% of checks are unnecessary in a suite of multithreaded benchmarks. On most of the benchmarks, identifying thread-local objects took longer than simply checking accesses on them. When I added the static filter, however, performance improved by 8% over the baseline.

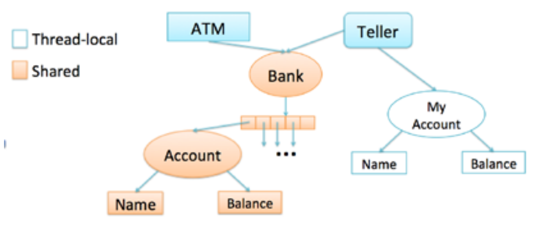

Figure 1: The account is initially thread-local because it is only reachable from the "Teller" thread.

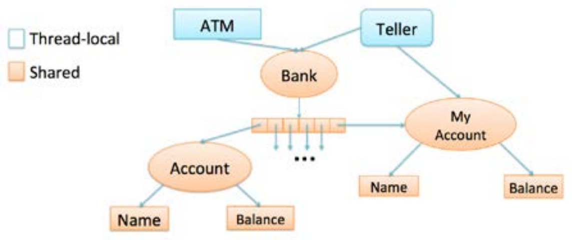

Figure 2: When the teller writes the account to the bank, it becomes reachable from the "ATM" and "Teller" threads. It is thus shared and must be tracked by the race detector.

More Information

For more information see: http://www.cs.williams.edu/~freund/races/index.html

About the Author

Emma Harrington is a senior at Williams College, where she studies computer science and economics. Next year she plans to go to graduate school in economics, where she will study the motivations of users contributing to online resources like StackOverflow. This past summer she was a Software Engineering intern at Fiksu, a mobile advertising company. In the summer of 2013, she worked with Professor

Freund on an independent research project, laying the groundwork for what is presented here. The summer after her freshman year, she co-wrote an article for Regulation, called “Scalping Scalpers or Consumers?” Outside of the lab, she is captain of the Williams College softball team and gives pitching lessons to local high-schoolers.
Email: Emma.Harrington@williams.edu