Perl For Computer Science Grad Students

Walt Mankowski (walt@cs.drexel.edu)
Drexel University

YAPC::NA
22 June 2010
Who am I?
CS grad student at Drexel
5 1/2 years
No, I don’t know when I’m going to defend
“Canonical Behavior Patterns”
What I thought CS grad would be like
"I think you should be more explicit here in Step Two."

\[
\begin{align*}
\Lambda & = 203 - 6.24 \\
\frac{2K^2}{5} & = \frac{0.06518}{0.04} \\
& + \frac{1}{10} \frac{345}{\sqrt{10}}
\end{align*}
\]
LISP is over half a century old and it still has this perfect, timeless aura about it.

I wonder if the cycles will continue forever.

A few coders from each new generation rediscovering the LISP arts.

These are your father's parentheses.

Elegant weapons for a more... civilized age.
What CS grad school is really like
PROCESS THE DATA USING METHOD X.

REALLY?

Yeah, remember? I spent weeks on it because you insisted it should work.

HMM.

Should I try it again?

No, no. That would be a waste of effort.

I'll ask one of the other students to do it.

I'm... actually ok with that.
Math, CS stuff
Finding Canonical Behaviors in User Protocols

Walter C. Mankowski, Peter Bogunovich, Ali Shokoufandeh, and Dario D. Salvucci
Department of Computer Science, Drexel University
Philadelphia, PA 19104, USA
{walt, pjb38, ashokouf, salvucci}@cs.drexel.edu

ABSTRACT
While the collection of behavioral protocols has been common practice in human-computer interaction research for many years, the analysis of large protocol data sets is often extremely tedious and time-consuming, and automated analysis methods have been slow to develop. This paper proposes an automated method of protocol analysis to find canonical behaviors — a small subset of protocols that is most representative of the full data set, providing a reasonable “big picture” view of the data with as few protocols as possible. The automated method takes advantage of recent algorithmic developments in computational vision, modifying them to allow for distance measures between behavioral protocols. The paper includes an application of the method to web-browsing protocols, showing how the canonical behaviors found by the method match well to sets of behaviors identified by expert human coders.

Author Keywords
Protocol analysis, sequential data analysis, web browsing

ACM Classification Keywords
H.1.2 Models and Principles: User/Machine Systems; H.5.2 Information Interfaces and Presentation: User Interfaces

INTRODUCTION
In the study of user behavior, researchers and practitioners alike often collect data in the form of behavioral protocols — sequences of actions recorded during the execution of a task. The analysis of these protocols provides a wealth of information about user behavior, and thus protocol data have been collected for a wide range of data types and utilized in a wide variety of ways. For instance, protocols have been employed for examining manual actions such as mouse clicks and keystrokes [e.g., 2], verbal reports [e.g., 7], and eye movements [e.g., 1], and sometimes combinations of these and other data [e.g., 9]. Based on this rich set of data, researchers have used protocols for such varied purposes as exploratory data analysis [e.g., 15], classification and recognition [14], and cognitive modeling [e.g., 17]. At the same time, the richness and quantity of protocol data have a severe limitation: it is typically difficult, if not impossible, to process and analyze the data by hand, thus requiring some form of data aggregation to make analysis and understanding feasible. While such aggregation provides information with respect to overall behavior, it washes over the interesting patterns that may appear in particular protocol instances from individual users.

In this paper we introduce the idea of canonical behaviors in user protocols and propose a computational method to identify them in an automated way. Canonical behaviors are a small subset of the protocol data set that is most representative of the entire data set, providing a reasonable “big picture” view of the data with as few protocols as possible. The identification of canonical behaviors is often performed laboriously by hand in standard protocol analysis; for instance, eye-movement or verbal protocol analysis often includes the identification and dissection of a few individual protocols that exemplify interesting strategies in the task [see, e.g., 7, 14]. While methods for automated protocol analysis have been studied in previous efforts, the methods focus on aligning observed protocols with the predicted behaviors of a user model [e.g., 13, 16]. Our work takes a very different but complementary approach, using a specification of the similarity between behaviors to automatically identify canonical behaviors in a set of user protocols.

To illustrate our approach in a real-world domain, we apply the method to the domain of web browsing. Recent efforts [e.g., 4, 5] have analyzed web-browsing behavior in a number of ways, typically involving some type of aggregation across individual user protocols. There has been some work on the analysis of individual protocols by aligning browsing protocols with the predictions of a cognitive or task model [3, 11, 12]. We aim to complement this work by proposing a method for finding canonical web-browsing behaviors without the need for any type of a priori model. The resulting canonical behaviors could be useful, for instance, in determining standard paths to desired information on a given web site, or in identifying circuitous paths of confused users and revising the web site accordingly.

FINDING CANONICAL BEHAVIORS
Our technique for computing canonical behaviors derives from work in the area of computational vision, where techniques have been developed to identify canonical members of a class of visual patterns [6]. In the context of user proto-
Computing the Canonical Subsets of User Protocols

Walter Mankowski, Peter Bogunovich, Ali Shokoufandeh, and Dario Salvucci
Vision and Cognition Laboratory, Drexel University, Philadelphia, PA, USA

Introduction

• We are interested in automatically finding canonical behaviors — a small subset of user protocols that is most representative of the full data set.
• The collection of user protocols is common in human factors research, but analyzing the large datasets produced can be tedious and time-consuming.
• Our canonical set algorithm can automatically identify canonical behaviors with no a priori model; all that is needed is a similarity measure between pairs of protocols.

Canonical Behaviors

Small subset of protocols which
a) is most representative of the entire set, and
b) contains as few protocols as possible
Problems:
a) Too much data to analyze by hand
b) Data aggregation often necessary
c) Interesting patterns can be missed

Canonical Sets

Denton’s approximation algorithm automatically finds the most representative samples from a set of patterns. Finding the canonical set can be expressed as an optimization problem, where the goal is to minimize the weights of the intra edges while simultaneously maximizing the weights of the cut edges. This is known to be NP-Complete, so we employ an approximation algorithm (Denton, 2008):

1. Formulate as integer programming problem
2. Relax to semidefinite or quadratic program
3. Use off-the-shelf solver in MATLAB

There is one free parameter, $\lambda \in [0, 1]$, which scales the weighting given to cut edges versus intra edges.

Web Browsing Experiment

• Each subject was asked 32 questions about a college website (www.cmu.edu)
  - No search engines permitted
  - 12 users (2 female, 10 male) unfamiliar with site
  - User protocol = series of pages visited by each user to answer each question
  • Found canonical protocols, then compared with “ground truth” — 2 expert human coders who identified distinct behaviors by hand for each question, using their own judgement of the best division of “similar” and “different” behaviors
  • Similarity measures:
    - edit distance (Mankowski et al., CHI 2009)
    - similarity of the induced subgraphs (Mankowski et al., GbR 2009)

Discussion

We have presented an automated method for finding canonical subsets of user protocols that matches well with those found by expert human coders. Potential uses include:

• Extracting critical protocols to facilitate the development of cognitive models of user behavior
• Analyzing a corpus of protocols by associating each behavior to its most similar canonical behavior

Future Work

• Development of similarity measures to support multi-modal data
• Development of stability measures suited to behavioral data, e.g. incorporating the timing of the actions in the protocol
• Larger-scale experiments

Results
LAST NIGHT I DRIFTED OFF WHILE READING A LISP BOOK.

Huh?

SUDDENLY, I WAS BATHED IN A SUFFUSION OF BLUE.

AT ONCE, JUST LIKE THEY SAID, I FELT A GREAT ENLIGHTENMENT. I SAW THE NAKED STRUCTURE OF LISP CODE UNFOLD BEFORE ME.

MY GOD
IT'S FULL OF CAR'S

THE PATTERNS AND METAPATTERNS DANCED.
SYNTAX FADED, AND I SWAM IN THE PURITY OF QUANTIFIED CONCEPTION. OF IDEAS MANIFEST.

TRULY, THIS WAS THE LANGUAGE FROM WHICH THE GODS WROUGHT THE UNIVERSE.

No, it's not.

It's not?

I MEAN, OSTensibly, YES. HONESTLY, WE HACKED MOST OF IT TOGETHER WITH PERL.
No one else in the department uses Perl
What do they use?
Matlab
Java
Python
C++
What’s this talk about?
Perl as a glue language
When to use perl?
When not to use perl?
Lessons learned
What do grad students do?
HOW GRAD SCHOOL IS JUST LIKE KINDERGARTEN

ALL DAY NAPPING IS ACCEPTABLE / THERE IS CONSTANT ADULT SUPERVISION

YOU GET COOKIES FOR LUNCH / MOST COMMON ACTIVITY: CUTTING AND PASTING

THERE ARE NO GRADES (YOU JUST HAVE TO PLAY WELL WITH OTHERS) / CRYING FOR YOUR MOMMY IS NORMAL

WWW.PHDCOMICS.COM
1. take classes
2. do research
3. etc.
Intro to Computer Graphics
Program Generation and Optimization
Mini MMM Performance (64 x 64 Matrices)
Mflops vs NB

- naive ijk
- blocking
- blocking+unrolling, k=2
- blocking+unrolling, k=4, ijk
- blocking+unrolling, k=4, jik
Mini MMM Performance (256 x 256 Matrices)

Mflops vs NB

naive ijk
blocking
blocking+unrolling, k=2
blocking+unrolling, k=4, ijk
blocking+unrolling, k=4, jik
Mini MMM Performance (512 x 512 Matrices)
Mflops vs NB

- naive ijk
- blocking
- blocking+unrolling, k=2
- blocking+unrolling, k=4, ijk
- blocking+unrolling, k=4, jik
set term postscript color
set xlabel 'NB
set ylabel 'Mflops'
set title "Mini MMM Performance (64 x 64 Matrices)\nMflops vs NB"
set grid
set key bottom
set xtics 16,4,80
set out "nb_mflops.64.eps"
plot 'nb_mflops.64.out' u 1:2 t 'naive ijk' w l
set out "nb_mflops.64.eps"
replot 'nb_mflops.64.out' u 1:3 t 'blocking' w l
set out "nb_mflops.64.eps"
replot 'nb_mflops.64.out' u 1:4 t 'blocking+unrolling, k=2' w l
set out "nb_mflops.64.eps"
replot 'nb_mflops.64.out' u 1:5 t 'blocking+unrolling, k=4, ijk' w l
set out "nb_mflops.64.eps"
replot 'nb_mflops.64.out' u 1:6 t 'blocking+unrolling, k=4, jik' w l
set term postscript color
set xlabel 'NB'
set ylabel 'Mflops'
set title "Mini MMM Performance (64 x 64 Matrices)\nMflops vs NB"
set grid
set key bottom
set xtics 16,4,80

set out "nb_mflops.64.eps"
plot 'nb_mflops.64.out' u 1:2 t 'naive ijk' w l

set out "nb_mflops.64.eps"
replot 'nb_mflops.64.out' u 1:3 t 'blocking' w l

set out "nb_mflops.64.eps"
replot 'nb_mflops.64.out' u 1:4 t 'blocking+unrolling, k=2' w l

set out "nb_mflops.64.eps"
replot 'nb_mflops.64.out' u 1:5 t 'blocking+unrolling, k=4, ijk' w l

set out "nb_mflops.64.eps"
replot 'nb_mflops.64.out' u 1:6 t 'blocking+unrolling, k=4, jik' w l
set term postscript color
set xlabel 'NB'
set ylabel 'Mflops'
set title "Mini MMM Performance (64 x 64 Matrices)\nMflops vs NB"
set grid
set key bottom
set xtics 16,4,80

set out "nb_mflops.64.eps"
plot 'nb_mflops.64.out' u 1:2 t 'naive ijk' w l

set out "nb_mflops.64.eps"
replot 'nb_mflops.64.out' u 1:3 t 'blocking' w l

set out "nb_mflops.64.eps"
replot 'nb_mflops.64.out' u 1:4 t 'blocking+unrolling, k=2' w l

set out "nb_mflops.64.eps"
replot 'nb_mflops.64.out' u 1:5 t 'blocking+unrolling, k=4, ijk' w l

set out "nb_mflops.64.eps"
replot 'nb_mflops.64.out' u 1:6 t 'blocking+unrolling, k=4, jik' w l
set term postscript color
set xlabel 'NB
set ylabel 'Mflops'
set title "Mini MMM Performance (64 x 64 Matrices)\nMflops vs NB"
set grid
set key bottom
set xtics 16,4,80
set out "nb_mflops.64.eps"
plot 'nb_mflops.64.out' u 1:2 t 'naive ijk' w l
set out "nb_mflops.64.eps"
replot 'nb_mflops.64.out' u 1:3 t 'blocking' w l
set out "nb_mflops.64.eps"
replot 'nb_mflops.64.out' u 1:4 t 'blocking+unrolling, k=2' w l
set out "nb_mflops.64.eps"
replot 'nb_mflops.64.out' u 1:5 t 'blocking+unrolling, k=4, ijk' w l
set out "nb_mflops.64.eps"
replot 'nb_mflops.64.out' u 1:6 t 'blocking+unrolling, k=4, jik' w l
set term postscript color
set xlabel 'NB
set ylabel 'Mflops'
set title "Mini MMM Performance (256 x 256 Matrices) Mflops vs NB"
set grid
set key bottom
set xtics 16,4,80

set out "nb_mflops.256.256_out" u 1:2 t 'naive ijk' w l

set out "nb_mflops.256.256_out" u 1:3 t 'blocking' w l

set out "nb_mflops.256.256_out" u 1:4 t 'blocking+unrolling, k=2' w l

set out "nb_mflops.256.256_out" u 1:5 t 'blocking+unrolling, k=4, ijk' w l

set out "nb_mflops.256.256_out" u 1:6 t 'blocking+unrolling, k=4, jik' w l
#!/usr/bin/perl -w
use strict;
use autodie;

my $n = $ARGV[0];
my $datafile = "nb_mflops.$n.out";
my $outfile = "nb_mflops.$n.eps";

open my $GP, "|-", "gnuplot";

print $GP <<EOT;
set term postscript color
set xlabel 'NB
set ylabel 'Mflops'
set title "Mini MMM Performance ($n x $n Matrices)\nMflops vs NB"
set grid
set key bottom
set xtics 16,4,80

set out "$outfile"
plot '$datafile' u 1:2 t 'naive ijk' w l
set out "$outfile"
replot '$datafile' u 1:3 t 'blocking' w l
set out "$outfile"
replot '$datafile' u 1:4 t 'blocking+unrolling, k=2' w l
set out "$outfile"
replot '$datafile' u 1:5 t 'blocking+unrolling, k=4, ijk' w l
set out "$outfile"
replot '$datafile' u 1:6 t 'blocking+unrolling, k=4, jik' w l
EOT
#!/usr/bin/perl -w
use strict;
use autodie;

my $n = $ARGV[0];
my $datafile = "nb_mflops.$n.out";
my $outfile = "nb_mflops.$n.eps";

open my $GP, "|-", "gnuplot";

print $GP <<EOT;
set term postscript color
set xlabel 'NB
set ylabel 'Mflops'
set title "Mini MMM Performance ($n x $n Matrices)\nMflops vs NB"
set grid
set key bottom
set xtics 16,4,80
set out "$outfile"
plot '$datafile' u 1:2 t 'naive ijk' w l
set out "$outfile"
replot '$datafile' u 1:3 t 'blocking' w l
set out "$outfile"
replot '$datafile' u 1:4 t 'blocking+unrolling, k=2' w l
set out "$outfile"
replot '$datafile' u 1:5 t 'blocking+unrolling, k=4, ijk' w l
set out "$outfile"
replot '$datafile' u 1:6 t 'blocking+unrolling, k=4, jik' w l
EOT
#!/usr/bin/perl -w
use strict;
use autodie;

my $n = $ARGV[0];
my $datafile = "nb_mflops.$n.out";
my $outfile = "nb_mflops.$n.eps";

open my $GP, "|-", "gnuplot";

print $GP <<EOT;
set term postscript color
set xlabel 'NB
set ylabel 'Mflops'
set title "Mini MMM Performance ($n x $n Matrices)\nMflops vs NB"
set grid
set key bottom
set xtics 16,4,80

set out "$outfile"
plot '$datafile' u 1:2 t 'naive ijk' w l
set out "$outfile"
replot '$datafile' u 1:3 t 'blocking' w l
set out "$outfile"
replot '$datafile' u 1:4 t 'blocking+unrolling, k=2' w l
set out "$outfile"
replot '$datafile' u 1:5 t 'blocking+unrolling, k=4, ijk' w l
set out "$outfile"
replot '$datafile' u 1:6 t 'blocking+unrolling, k=4, jik' w l
EOT
#!/usr/bin/perl -w
use strict;
use autodie;

my $n = $ARGV[0];
my $datafile = "nb_mflops.$n.out";
my $outfile = "nb_mflops.$n.eps";

open my $GP, "|-", "gnuplot";

print $GP <<EOT;
set term postscript color
set xlabel 'NB
set ylabel 'Mflops'
set title "Mini MMM Performance ($n x $n Matrices)\nMflops vs NB"
set grid
set key bottom
set xtics 16,4,80
set out "$outfile"
plot '$datafile' u 1:2 t 'naive ijk' w l
set out "$outfile"
replot '$datafile' u 1:3 t 'blocking' w l
set out "$outfile"
replot '$datafile' u 1:4 t 'blocking+unrolling, k=2' w l
set out "$outfile"
replot '$datafile' u 1:5 t 'blocking+unrolling, k=4, ijk' w l
set out "$outfile"
replot '$datafile' u 1:6 t 'blocking+unrolling, k=4, jik' w l
EOT
see also
Chart::Gnuplot
see also
Chart::Clicker
SQLite
SQLite’s different...
embedded, in-process library
very small
very fast
great for grad student projects
simple types
(like in Perl)
simple security model
database is a file
check it into a vc
scp it to a faster box
email it to collaborators
Research projects
Very different from code for classes or jobs
Grad student research is about getting results and writing papers
The point is the results
Code is a way to get results
No one cares how beautiful your code is
No one cares how shitty your code is
Remember, this is research
No one’s ever done it before
Lots of things you try won’t work
Optimize for flexibility
This sounds a lot like agile programming, doesn’t it?
Raw data
Raw data

SQLite
Denton et al., 2004
Determining Canonical Set

1. Create complete graph where edge weights are similarity between the corresponding behaviors

2. Minimize weights of intra edges while simultaneously maximizing weights of cut edges

3. This optimization is known to be NP-Complete
Minimize  \[ \lambda_1 \left( \frac{1}{4} \sum_{i,j} W_{ij} (1 + y_i)(1 + y_j) \right) \]

+ \[ \lambda_2 \left( \frac{1}{2} \sum_{i,j} W_{ij} (y_i - y_j) \right) \]

+ \[ \lambda_3 \left( \frac{1}{2} \sum_{i=1}^{n} t_i (1 - y_i) \right) \]

Subject to  \[ \frac{1}{2} \sum_{i=1}^{n} (1 + y_i) - k_{\text{min}} \geq 0, \]

\[ k_{\text{max}} - \frac{1}{2} \sum_{i=1}^{n} (1 + y_i) \geq 0, \]

\[ y_i \in \{-1, +1\}, \quad \forall 1 \leq i \leq n \]
quadprog()
(in Optimization Toolbox)
Running Matlab from Perl
$res = `matlab -r ...`
use Expect.pm to automate Matlab session
Math::Matlab
Earth Movers Distance
Other useful modules
Graph.pm
Math::MatrixReal
Set::Scalar
TO PHD OR NOT TO PHD...

THAT IS THE QUESTION.

WHETHER 'TIS SANER IN THE MIND TO SUFFER THE SLIGHTS AND PUT-DOWNS OF OUTRAGEOUS FACULTY...

OR TO TAKE DATA DESPITE ADVISOR GRUMBLING?

AND BY GRADUATING, END THEM?

TO GRADUATE: TO SLEEP; ONCE MORE; AND, BY A PHD TO SAY WE END THE BACKACHE AND THOUSAND FINANCIAL DEBTS THAT GRADS ARE HEIR TO.

TIS A COMMENCEMENT DEVOUTLY TO BE WISH'D.

TO GRADUATE, TO SLEEP...

TO NAP: PERCHANCE TO DREAM...

phd.stanford.edu
THANKS!
simulator (Beusmans & Rensink, 1995). The simulated environment was a three-lane highway in a construction zone with driving restricted to the center lane, as shown in Figure 1. The road alternated between segments of straight roadway and segments of various curvatures, all of which could be negotiated comfortably at highway speeds without braking. The driver followed three cars and was tailed by another car, which was visible in the simulated rear-view mirror. The speed of lead car varied from 5-35 m/s (11-78 mph) according to a sum of three sinusoids that resulted in an apparently random pattern. The rear car followed at distance of 9-21 m (29-68 ft) also varying as the sum of three sinusoids. Cones on either side of the center lane prevented drivers from passing other cars and emphasized the need for maintaining a central lane position. Thus, the cell-phone dialing scenario could be thought of in terms of the driver being caught up in a construction zone and needing to call several people to notify them of a delay.

For the dialing task, we employed a commercially-available cell phone (Samsung SCH-3500 with Sprint PCS), shown in Figure 2. This phone (like many similar phones) allows for multiple methods of dialing. In order to examine differential effects of various dialing methods, we chose four of the phone's built-in methods, which can be described as follows:

- **Manual**: dial the phone number and press Talk
- **Speed**: dial the party's single-digit “speed number” and press Talk
- **Menu**: press the up arrow to access menu, scroll down to the desired party with the down arrow, and press Talk
- **Voice**: press and hold Talk, say the party's name when prompted, and wait for confirmation

Table 1 shows examples of using each of these dialing methods to make a call. Note that two of the methods, speed and menu dialing, require that numbers be added to an internal phone book and associated with a unique “speed number.” The four methods thus serve well to illustrate our modeling approach for comparing effects of different dialing methods on driver performance.

**The Integrated Dialing-Driving Model**

The prediction of effects of dialing on driving centers on an integrated cognitive model that combines individual models for each task. To facilitate the development and integration of these models, we implemented the models in the ACT-R cognitive architecture (Anderson & Lebiere, 1998). ACT-R is a production-system architecture based on condition-action rules that execute the specified actions when the specified conditions are met. Like most cognitive architectures, ACT-R provides a rigorous framework for cognitive models as well as a set of built-in parameters and constraints on cognition and perceptual-motor behavior (when using ACT-R/PM: Byrne & Anderson, 1998); the parameters facilitate a priori predictions about behavior, while the constraints facilitate more psychologically (and neurally) plausible models. In addition, the architecture allows for straightforward integration of models of multiple tasks: generally speaking, the modeler can combine the models' rule sets and modify the rules to interleave the multiple tasks (see Salvucci, 2001). All these qualities of the architecture are essential to our ability to integrate models of dialing and driving to predict the effects of each task on the other.

**Dialing Models**

We first consider the development of the cognitive models for dialing the cell phone using each of the four methods. To this end we employed a straightforward task analysis and implemented a simple, minimal model for each method based on this analysis. The procedure required by the cell phone highly constrains the model in that it specifies the sequence of keypresses needed to dial. However, the model must also incorporate the cognitive and perceptual processes needed to execute the procedure.