Presentation of Perracotta: Mining Temporal API Rules from Imperfect Traces

Tao Xia

October 19 2006

Motivation

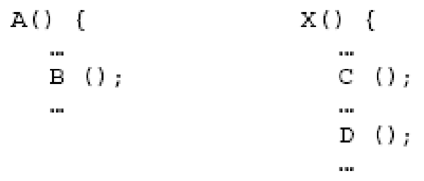
- Dynamic inference technique working with
  - imperfect traces
  - large scale of software
  - filtering uninteresting properties

Satisfaction Rate

- Partition the trace into sub-traces
- P(AL) = n(AL)/n
  - n(AL) the number of partitions satisfy the Alternating template
  - n the total number of partitions

Properties Selection

- Reachability



```
A() {                  X() {
    ...                    ...
    B ();                  C ();
    ...                    ...
                           D ();
                           ...
```

Figure 3. Reachable and unreachable events

- Name Similarity
  - 2w/(wP+wS)
  - w is the common word between P and S
- Chaining
  - If A>B, B>C, and A>C then it form a chain A>B>C

Results

- Satisfaction threshold for JBoss and Wir

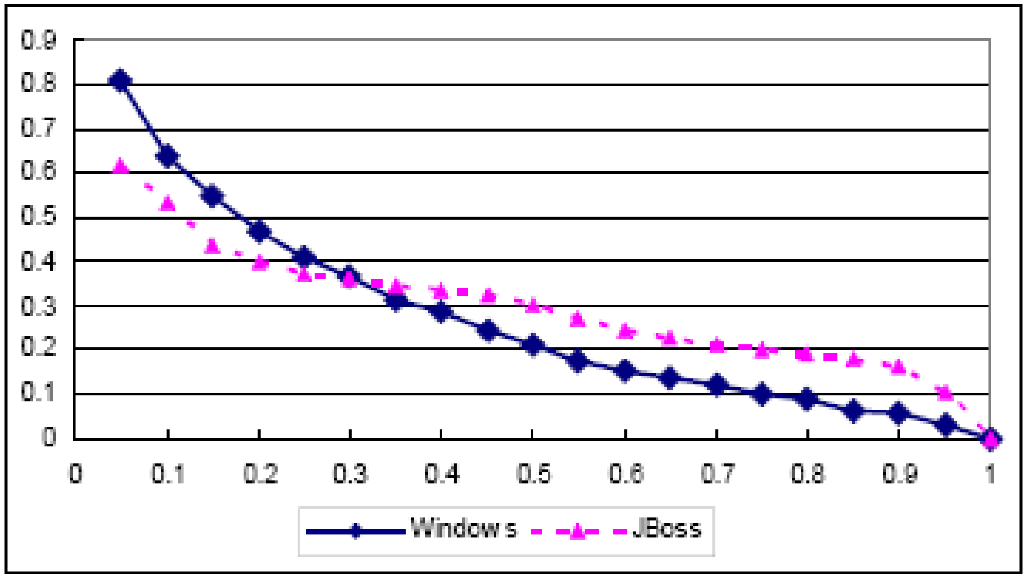
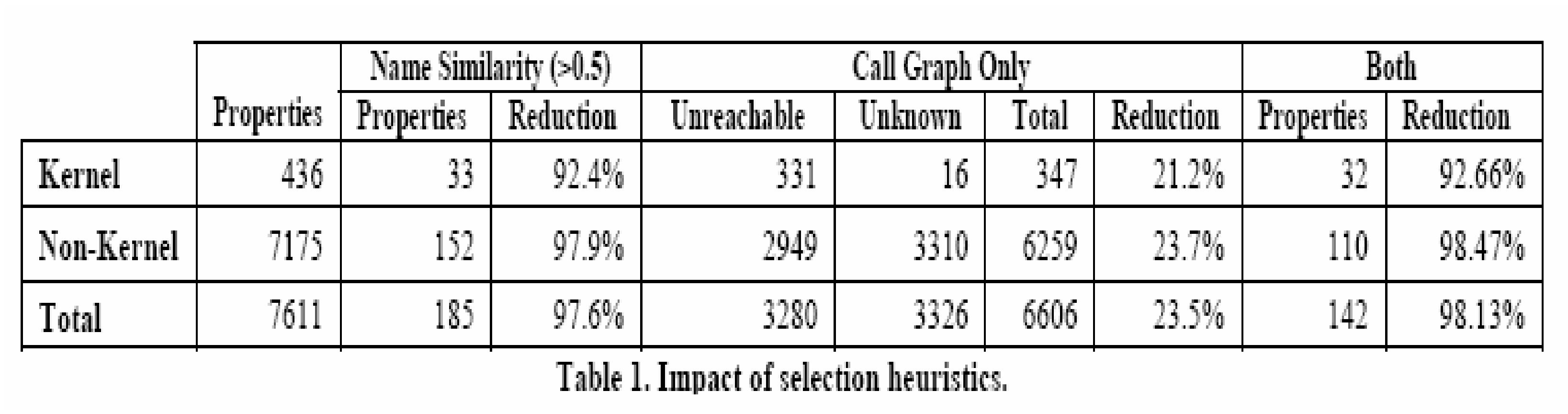


Figure 4. Inferred properties versus satisfaction threshold. The horizontal axis is the threshold varying from 0.0 to 1.0. The vertical axis is the fraction of properties that above the threshold.

Results

- Properties Selection for Windows



	Properties	Name Similarity (>0.5)		Call Graph Only				Both	
		Properties	Reduction	Unreachable	Unknown	Total	Reduction	Properties	Reduction
Kernel	436	33	92.4%	331	16	347	21.2%	32	92.66%
Non-Kernel	7175	152	97.9%	2949	3310	6259	23.7%	110	98.47%
Total	7611	185	97.6%	3280	3326	6606	23.5%	142	98.13%

Table 1. Impact of selection heuristics.

Feedback

- Positive
  - Works with imperfect traces
  - Good performance again large scale software
  - Found a serious bug in Windows
- Negative
  - It would be interesting to see how other property patterns work (Eight property pattern)
  - Implementation section is confusing