GI++ == Focused Auto Programming?

Robert Feldt
Chalmers University of Technology, Sweden
at the COW-50, UCL, London, 2017-01-31

@drlfeldt
One view of SBSE: Ever-expanding Success!
A contrarian view of SBSE: Not quite there yet...
A contrarian view of SBSE: Not quite there yet...
"Evolution is the natural way to fly on a plane running software evolved by a program like this, than fly on a plane running software I wrote myself," says Hillis, programmer extraordinaire.
In his 1950 paper “Computing Machinery and Intelligence,” Turing described how evolution and natural selection might be used to automatically create an intelligent computer program [2].

“We cannot expect to find a good child-machine at the first attempt. One must experiment with teaching one such machine and see how well it learns. One can then try another and see if it is better or worse. There is an obvious connection between this process and evolution, by the identifications
“Structure of the child machine = Hereditary material”
“Changes of the child machine = Mutations”
“Natural selection = Judgment of the experimenter”

[Koza2010] in GPEM Anniversary issue
Some common GP/SBSE “cop outs”

- Tune only constants/numbers in fixed program
- Delete/remix existing code
- Focus on (minimal) interfaces between existing codes
- Focus on non-mainstream/obscure languages / processing formalisms where humans (currently) have less experience
- Evolve test data rather than programs
- Evolve test cases and not programs
- Requiring lots and lots of example Input/Outputs
- ...

Clear goal, small search space,
less/short structure
A continuum of Automated Programming
Focused Automated Programming

• I propose we should study FAP! aka…
  • Domain-specific Automated Programming (DAP)
  • Task-specific Automated Programming (TAP)
  • Defined as: “Focused application of search and optimisation to create/adapt/tune (parts of) program code during its development, setup and/or execution”
  • Focused here essentially means “human-guided”, i.e. it is a hybrid/interactive development philosophy
  • => we need ideas, intuition and methods/processes for how to use search/optimisation more actively in the software development process
{
  "name": "V Basili",
  "citations": 33501,
  "h-index": 82
}
Web extraction, traditional solution vs AdaptiLib

- WebGet Lib
- XML Parser Lib
- Regex Lib
- Custom code

- AWE Lib
- Examples
Adaptive Libraries

- A normal library (lib):
  - 1. has a number of functions that can be called
  - 2. to solve specific tasks
  - 3. has documentation to describe the functions
  - 4. and examples to understand API & how to put together
  - But only 1 above is directly useable without a human
  - 2-4 requires a human to assemble solution based on text

- Adaptive libraries (AdaptiLibs):
  - 1. Still has basic “atoms” = functions to be called
  - (2a) But also executable examples that uses atoms to perform specific, named sequences
  - (2b) And allow fuzzy mapping of user needs to tasks
Example: Adaptive Web Extraction (AWE!) library, in practice

elements = [
    ("scholar.google.se/citations?user=B3C4aY8AAAAJ&hl=en",
    {"name": "V Basili",
       "citations": 33501,
       "h-index": 82}),
    ("scholar.google.se/citations?user=Zj897NoAAAAJ&hl=en",
    {"name": "Lionel Briand",
       "citations": 21505,
       "h-index": 69})]

gscholar_ex = create_extractor(examples)

extract(gscholar_ex, "scholar.google.se/citations?user=CQDOm2gAAAAJ&hl=en")

# returns:
# {"name": "Barbara Ann Kitchenham",
# "citations": 63,
# "h-index": 154})
Big benefits with semantically similar task

Only change 2 I/O examples & re-adapt!
GI would not help: Only semantic, not syntactic similarity

"...:<a>Citations</a></td><td class="gsc_rsb_std">33501</td><td class="gsc_rsb_std">9054</td>...

"...:{"hIndex":51,"estimatedTotalCitationCount":{"min":31675,"value":36839,"max":42905,..."}
Design Rules for AdaptiLibs (so far…)

- Start by defining basic “atomic” operations
  - Type conversion operations: parseToInt, parseToFloat
  - Data transformation: uppercase, lowercase, leadingcase
  - Basic data access: get_url
  - Matching: matchregexp, matchregexp_ignorecase
- Go through concrete task from example & note how a human solves it in as atomic steps as possible
- Extend with atoms, and possibly (complex) atom seq.
- Feldt’s Law for Designing Lib incl. Search, consider in order:
  - 1. Deterministic / Exact (fastest, most efficient)
  - 2. Heuristics / Approximations (order by applicability)
  - 3. Focused Search (part of solution only, then aggregate)
  - 4. Interact / Ask Developer (in adapt step)
  - 5. Full/free search (search from atoms & up, warn dev)
Conclusions

- Despite many promises of GP & SBSE it has under delivered on practical Automated Programming
- Compared to other SBSE, GI comes closer to AP
- As techniques and processing power increase we will see more practical AP
- But semantic similarity does not imply syntactic similarity
  => less opportunity for detailed code reuse
- But we can also deliver practical AP now by hybridising it with human intelligence and guidance
- We are developing AdaptiLibs, general libraries that adapt to I/O examples of users/developers
- Combines task-driven design & experience of humans
- with brute force and flexibility of search, only wh. needed
Thank you!

robert.feldt@chalmers.se

@drfeldt
But what about Bartoli et al?!

| Extraction task $E_0$                  | $|E_0|$ | $\sum_{E_0} \ell(s)$ | $\sum_{E_0} |X_s|$ | $\sum_{E} |X_s|$ | LR  | Fm  | Prec | Rec | Fm  | EC  | TtL |
|---------------------------------------|--------|----------------------|----------------------|----------------------|-----|-----|------|-----|-----|-----|-----|
| ReLIE-Web/All-URL                    | 3,877  | 4,240                | 502                  | 24                   | 5.0 | 99.2| 90.0 | 91.9 | 90.9 | 2.6 | 15  |
|                                       | 50     | 10.0                 | 99.2                 | 92.1                 | 95.0 | 93.5| 6.4  | 35  |
|                                       | 100    | 19.9                 | 98.9                 | 94.8                 | 96.5 | 95.6| 13.7 | 71  |
| ReLIE-Web/HTTP-URL                   | 3,877  | 4,240                | 499                  | 24                   | 5.0 | 99.2| 86.3 | 89.0 | 87.6 | 2.5 | 11  |
|                                       | 50     | 10.0                 | 99.0                 | 91.0                 | 93.3 | 92.2| 5.8  | 32  |
|                                       | 100    | 20.0                 | 98.8                 | 92.9                 | 96.8 | 94.8| 13.1 | 66  |
| ReLIE-Email/Phone-Number              | 41,832 | 8,805                | 5,184                | 24                   | 0.5 | 97.7| 37.1 | 92.6 | 48.3 | 3.4 | 8   |
|                                       | 50     | 1.0                  | 99.0                 | 29.9                 | 96.6 | 43.3| 6.0  | 16  |
|                                       | 100    | 1.9                  | 98.9                 | 22.7                 | 98.3 | 35.8| 14.4 | 39  |
| Cetinkaya-HTML/href                   | 3,425  | 154                  | 214                  | 24                   | 11.7| 100.0| 98.7 | 99.2 | 98.9 | 2.5 | 12  |
|                                       | 50     | 23.4                 | 100.0                | 98.1                 | 98.7 | 98.4| 4.9  | 26  |
|                                       | 100    | 46.7                 | 99.8                 | 98.4                 | 99.1 | 98.8| 9.0  | 59  |
| Cetinkaya-HTML/href-Content*          | 3,425  | 154                  | 214                  | 24                   | 11.7| 98.4 | 74.9 | 98.7 | 80.6 | 2.4 | 16  |
|                                       | 50     | 23.4                 | 98.5                 | 85.1                 | 98.8 | 88.2| 4.8  | 29  |
|                                       | 100    | 46.7                 | 98.5                 | 83.2                 | 96.8 | 86.2| 10.5 | 67  |
| Cetinkaya-Web/All-URL                 | 1,234  | 39                   | 168                  | 24                   | 14.9| 99.2 | 99.4 | 98.8 | 99.1 | 1.7 | 3   |
|                                       | 50     | 29.8                 | 100.0                | 95.5                 | 98.6 | 96.9| 3.2  | 8   |
|                                       | 100    | 59.5                 | 99.5                 | 98.8                 | 98.8 | 98.8| 5.2  | 16  |
| Twitter/Hashtag+Citation              | 50,000 | 4,344                | 56,994               | 24                   | 0.1 | 100.0| 98.8 | 100.0| 99.4 | 1.2 | 3   |
|                                       | 50     | 0.1                  | 99.6                 | 99.2                 | 100.0| 99.6| 2.2  | 4   |
|                                       | 100    | 0.2                  | 99.8                 | 99.0                 | 100.0| 99.5| 4.6  | 7   |
| Twitter/All-URL                       | 50,000 | 4,344                | 14,628               | 24                   | 0.2 | 100.0| 94.7 | 98.5 | 96.6 | 1.8 | 3   |
|                                       | 50     | 0.3                  | 100.0                | 96.2                 | 98.3 | 97.2| 3.4  | 8   |
|                                       | 100    | 0.7                  | 99.4                 | 96.1                 | 98.0 | 97.0| 7.7  | 16  |
| Twitter/Username*                     | 50,000 | 4,344                | 42,352               | 24                   | 0.1 | 100.0| 99.3 | 100.0| 99.7 | 1.2 | 2   |
|                                       | 50     | 0.1                  | 100.0                | 99.2                 | 100.0| 99.6| 2.2  | 2   |
|                                       | 100    | 0.2                  | 99.9                 | 99.3                 | 100.0| 99.7| 4.6  | 2   |