ControlFlag: A Self-supervised Idiosyncratic Pattern Detection System for Software Control Structures

Niranjan Hasabnis Justin Gottschlich

Machine Programming Research Lab, Intel Labs, Intel

ControlFlag: flags idiosyncratic pattern violations

ControlFlag is a **Machine Programming** system that learns idiosyncratic patterns from open-source repos and uses them to flag idiosyncratic pattern violations.

Uncommon patterns in C language	Common patterns in C language	Type of problem
`if (x = 7)`	`if (x == 7)` `if (x != 7)`	Typographical error detection
`if (x->f)`	`if (x != NULL && x->f)` `if (x && x->f)`	Missing NULL check

Table: Examples of common and uncommon patterns in C language

Such knowledge can enable us to provide immediate feedback to developers in a live programming environment (IDE).

Can't compilers/static analyzers catch this case?

Yes, GCC-10.2 with `-Wall` can warn in case of an assignment in if statement as below

```
test.cpp:3:9: warning: suggest parentheses around assignment used as truth
                      value [-Wparentheses]
        if (x = 7) y = x;
```

Compilers and static analyzers have limitations:

- **rules-based approach** that is **labor-intensive** and
- seems **difficult to use in live-programming environment** (where complete program is generally not available).

ControlFlag can learn such rules automatically from vast amount of open-source code.

Formulating the problem as anomaly detection problem

Hypothesis: *certain patterns are uncommon in the control structures of high-level languages*

Approach:

1. **Mine** idiosyncratic patterns
2. **Check** user's patterns against the mined patterns.
3. **Flag** those that deviate as per anomaly threshold.

Contributions

- The **first-of-its-kind approach** (to our knowledge) that is **self-supervised** in its ability to learn idiosyncratic patterns and apply them to flag anomalous patterns.
- Preliminary implementation for C/C++ **mines 38M patterns** from `if` statements of **6000 GitHub repos** (having more than 100 stars) and **1B lines of code**.
- The anomaly found by ControlFlag in CURL is acknowledged by the developers and is fixed promptly. **This is the first example of ControlFlag's contribution to potentially improve robustness of real-world software**.

Cite as

Niranjan Hasabnis and Justin Gottschlich.
ControlFlag: A self-supervised idiosyncratic pattern detection system for software control structures.
34th Conference on Neural Information Processing Systems (NeurIPS 2020), ML for Systems Workshop, 2020.

ControlFlag: design

ControlFlag consists of two main phases: **pattern mining** and **scanning for idiosyncratic patterns**.

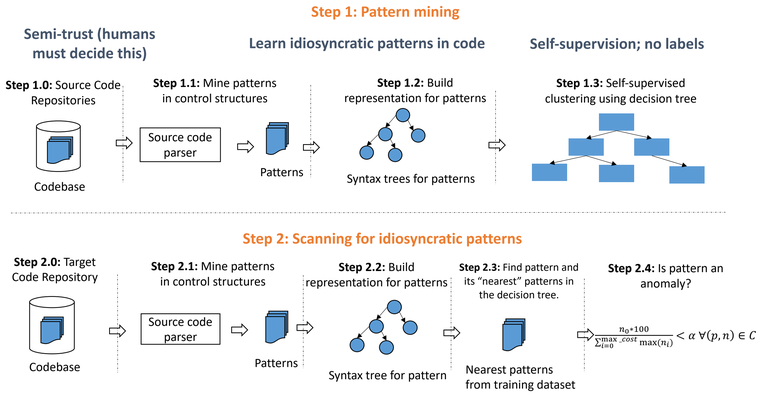


Figure: Overview of ControlFlag System.

Salient points of the design

Self-supervised learning system that uses semi-trusted training data

ControlFlag is a self-supervised learning system that uses GitHub repositories with more than 100 stars for mining patterns. Such a selection gives us semi-trust in the data.

Representations using levels of abstraction

Mined patterns are represented internally in ControlFlag as abstract syntax trees (referred as L1 abstraction level). They are also converted into higher-level tree representation (referred as L2 abstraction level) that drops certain details in ASTs that could reduce chances of finding the pattern in the training data.

Decision tree based clustering

Mined patterns in the tree forms are clustered together using a decision tree, which stores the patterns as well as their number of occurrences in the training data.

Suggesting auto-corrections for anomalous patterns

Target patterns are checked against the decision tree. A pattern is flagged as anomalous if it satisfies any of following: 1) it is not found in the decision tree, or 2) its occurrences are lower than the anomaly threshold defined below. ControlFlag suggests possible corrections of the anomalous pattern.

$$\frac{n_0 \times 100}{\sum_{i=0}^{max\_cost} max(n_i)} < \alpha \; \forall (p, n) \in C$$

C is the set of automatic correction results, in which every result contains a corrected pattern p and its occurrences n. α is a user-defined anomalous threshold, and max is a function that calculates the maximum of a list of occurrences. n is the number of occurrences of a pattern, with n_i being the number of occurrences of the pattern at i distance away.

Evaluation

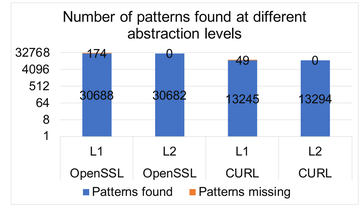
We used 6000 GitHub repositories (with more than 100 stars) for C language to mine patterns in `if` statement. This training data was obtained from 2.57M programs, 1.1BLoC and had 38M patterns.

AST	Occurrences	Example C expressions
`(identifier)`	4.3M	`if (x)`
`(unary_expr (``'!'') (identifier))`	2.09M	`if (!x)`
`(field_expr (identifier)(field_identifier))`	1.3M	`if (p->f)`
`(binary_expr (``'==') (identifier)(identifier))`	1.16M	`if (x == y)`
`(binary_expr (``'==') (identifier)(null))`	790K	`if (p == NULL)`
`(binary_expr (``'=') (identifier)(number)))`	487	`if (x = 0)`
`(binary_expr (``'=') (identifier)(identifier)))`	476	`if (x = y)`
`(binary_expr (``'%') (identifier)(number)))`	6468	`if (x % 2)`

Table: A table showing some of the most and least frequently occurring patterns at L1 abstraction level

Scanning for anomalous patterns in OpenSSL and CURL

We scanned OpenSSL and CURL open-source packages for anomalous patterns at the anomaly threshold of 1% and 5%.



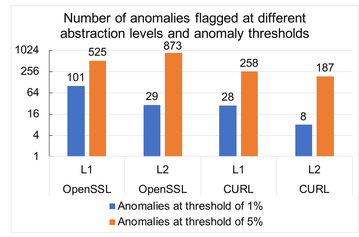


Figure: Results of scanning OpenSSL and CURL for anomalous patterns

CURL developers acknowledge and fix the flagged anomaly

Potential anomaly: `(s->keepon > TRUE)`

Location: `curl/lib/http_proxy.c:359`

Possible corrections:

`(s->keepon > TRUE)`, edit distance 0, occurrences 4

`(s->keepon > number)`, edit distance 2, occurrences 127540

`(s->keepon > variable)`, edit distance 2, occurrences 56475

Re: Potential confusion in http_proxy.c and a recommendation

- *This message:* [Message body] [More options (top, bottom)]
- *Related messages:* [Next message] [Previous message] [In reply to] [Next in thread] [Replies]
- *Contemporary messages sorted:* [by date] [by thread] [by subject] [by author] [by messages with attachments]

From: Daniel Stenberg via curl-library <curl-library_at_cool.haxx.se>
Date: Mon, 9 Nov 2020 23:51:20 +0100 (CET)

On Mon, 9 Nov 2020, Hasabnis, Niranjan via curl-library wrote:

> We believe that using "if (s->keepon > 1)" would eliminate this confusion
> and capture the intended semantics precisely.

I think you've pointed out code that could be written clearer, yes. But I think an even better improvement to this logic would be to use an enum or defined values that include all three used values as state names.

What do you think about my proposal over at:

https://github.com/curl/curl/pull/6193

Conclusion

- ControlFlag is a **self-supervised system** that learns idiosyncratic patterns in the control structures. It uses that knowledge to flag anomalous patterns that can potentially represent various problems.
- **Preliminary implementation flags anomalous pattern in CURL that is acknowledged by the developers and fixed promptly.** We believe that this is the first example of ControlFlag's contribution to potentially improve robustness of real-world software.