UCLouvain

École polytechnique de Louvain

LINFO2335
Programming paradigms

Report : Program Query Languages

Student :	*Noma :*
Brieuc Dubois	23752000

23 mai 2024

Table des matières

1 Introduction

In this report, we will expose the differences observed while testing multiple program query languages. We will first present the three languages we have chosen to test, then we will present the results of our experiments and finally, we will discuss the differences between the languages and our opinion about them.

2 Languages

RegEx is a pattern matching language that is widely used in text processing. It is a very powerful language but may also be very cryptic and hard to read. It is based on regular expressions. RegEx are available in many programming languages and tools, making it a very versatile language.

Pyttern is a pattern matching language that is based on Python. It allow more abstract patterns than RegEx and is generally easier to read. As it is based on the python synthax, it is also easier to learn.

AST Traversal isn't itself a pattern matching language, but a way to traverse the Abstract Syntax Tree of a program. As it is very versatile, it can be used to implement pattern matching.

3 False positives

While **RegEx** is very powerful to match strings in any scenario, it lacks some meta understanding of the data it is processing, such as variable names and make it quite difficult to match complex patterns. A typical example would be to match a variable or function name in a code. It should first ensure that the name isn't part of a comment, another name or a string. This is very hard to do with RegEx.

On the other hand, **Pyttern** is more powerful than RegEx and can match more complex patterns and has a better meta understanding of the code. It prevents most basic false positives.

AST Traversal is probably somewhere in between. It allow some understanding of the code, but is not as powerful as Pyttern. However, the risk of false positives is low, because the AST has already a knowledge of the code structure and

4 False negatives

Due to the strict nature of **RegEx**s, it is very easy to miss some patterns, in some particular cases, like the one mentioned above. It is also very hard to match patterns that are spread over multiple lines.

With **Pyttern**, while it's still possible to miss some patterns, it is way less likely to happen. Pyttern provide the degree of abstraction needed to match most of complex patterns.

AST Traversal is more controversial. If done correctly, it can match almost any pattern, no matter how it is presented, but due to the imbricated nature of the AST, it is also very easy to miss some patterns.

5 Learning curve

RegEx is probably the most complex to learn, due to it complex and not really easy to read syntax. But in the same time, it's the only one we had already experienced and used in the past.

AST Traversal needs some times to figure out how AST works, but once it's done, it's not very hard to use it, even for advanced patterns.

Pyttern is the easiest to learn, as it is based on Python and use very clear and simple syntax. It also abstract a lot of concepts, making it really easy to use.

6 Usability

RegEx is not easy to use, even with some experience. The syntax remains hard to read, write, understand and debug, and it's very easy to make mistakes without noticing it.

AST Traversal needs a lot of boilerplate code to work, and is more verbose compared to the other languages. But the code itself isn't very difficult to read or write.

Pyttern is the easiest to use, as it abstract most of the complexity. It's possible to write complex queries in a very short time, and the code is very easy to read.

7 Performance

RegEx is fast and efficient, and can be used in almost any language. It strict syntax makes it very fast to parse and execute.

Pyttern is also very fast, but may be slower than RegEx in some cases, due to the fact that it is based on Python, and that it is more abstract.

AST Traversal is the slowest of the three, as it needs to parse the AST before being able to match any pattern. It also requires the code to be correct, as it is based on the AST.

8 Output

Complex **RegEx** syntaxes can return some information about the matched pattern, but it is very hard to extract it. It is also very hard to extract the context of the match.

Pyttern is very powerful in this regard, as it can return the matched pattern, the context of the match, and even the position of the match in the code.

AST Traversal is also very powerful, as it can return the matched pattern and the context of the match. It can also return the AST node that matched the pattern for further processing.

9 Expressiveness

RegEx can produce very precise matches, with a high level of details. However, such queries are really complex and error-prone.

Pyttern is also very expressive, but is way easier to write and read. It can match complex patterns with a high level of details, while keeping the code simple.

AST Traversal is the most expressive of the three, as it can match almost any pattern, no matter how complex it is.

10 Declarative

RegEx describe what the pattern to match should look like, and let the language itself figure out how to match it.

Pyttern is also declarative, but is more abstract than RegEx, and can match more complex patterns.

AST Traversal is not declarative, as it needs to describe how to match the pattern, and not what the pattern should look like.

11 Tooling

RegEx is available in almost any language, making it usable in almost any context. It is also very well documented.

Pyttern is available only in Python, making it more specific and require a certain environment to be used. It is also less documented than RegEx, as it's less known.

AST Traversal is available in almost any language that has an AST, but is less known than RegEx. It can also adapt to almost any parser, making it very versatile and can be more specific on what matter in a specific context.

12 Conclusion

In conclusion, we can say that **RegEx** is a very powerful language, but is also very hard to use and understand. It is very fast and efficient, but can be very error-prone. **Pyttern** is a very good alternative to RegEx, as it is more powerful, easier to use and understand, and can match more complex patterns. **AST Traversal** is also a good alternative, as it can match almost any pattern, but is less known and less documented than the two others. It is also slower than the two others.