Context-Free Grep and Hierarchical Diff[1]

Gabriel A. Weaver and Sean W. Smith
Dartmouth College

Abstract

The rise in high-level programming languages in system in network administration has affected the utility of *grep* and *diff* when used on files written in these languages. We have observed this problem firsthand in our collaborations with network administrators at Dartmouth Computing and in the electric power industry. We are building *context-free grep* and *hierarchical diff* to address the limitations of *grep* and *diff* respectively. For both of our tools, we give examples of real-world problems they could address, sketch their design and evaluation, and describe their impact if built well.

1 Introduction

High-level languages for system and network administrators are increasingly on the rise over flat-file or line-based formats. Consider system configuration [13]. Tools such as CFEngine [2] and Puppet [9] allow system administrators to programmatically configure their systems. Even lower-level languages like Cisco IOS and NxOS have a hierarchical command structure; they are essentially programming languages.

In contrast, traditional *grep* and *diff* are geared towards the old file paradigms. *Grep* matches patterns of text, usually within lines, with regular expressions. Regular expressions, however, cannot recognize languages within parentheses nested arbitrarily deep. Therefore, regular expressions are not powerful enough to extract blocks of code in modern configuration languages.

Diff compares files line by line. Line by line comparison, however, is too low-level for modern configuration languages because line by line comparison ignores syntactic blocks. Syntactic blocks carry meaningful semantics of interest to practitioners. For example, practitioners may want to understand how a set of *network interfaces* changes. Traditional diff, however, will only report the set of *lines* (some of which pertain to interfaces) that change across multiple versions of a router configuration file.

2 Context-Free Grep and Hierarchical Diff

We rethink *grep* and *diff* in order to accommodate the rise in high-level programming languages and propose *context-free grep* and *hierarchical diff* respectively.

Context-Free Grep *Context-free grep* will let administrators casually extract "lightweight"[2] context-free structure just as readily as they currently extract regular structure using traditional *grep*.

Using *context-free grep*, a network administrator may extract blocks of Cisco IOS related to "security policy" within a given router (router.v1). In Figure 1, the administrator finds instances of the *service-policy* command within two of the router's interfaces. Our tool relies upon libraries of lightweight grammars defined for configuration languages, and outputs configuration blocks as parse trees. Although we have an implementation roadmap and prototype specifically for Cisco IOS, we have yet to implement the tool.

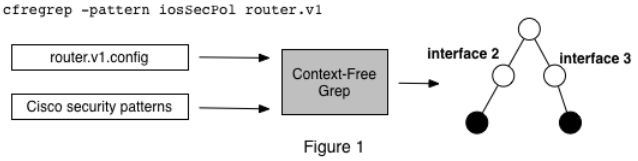

Figure 1

Hierarchical Diff *Hierarchical diff* will let administrators compare files in terms of the hierarchical structure of the language in which the file is written rather than by line number. For example, Cisco IOS router configurations, as well as other languages used by administrators, have a hierarchical structure. Although Cisco IOS lacks a formal grammar, we can still write a pseudo grammar for a meaningful subset of the language and compare parse trees. The process of writing grammars and recognizers for subsets of a given language also prevents differential parse tree attacks [7].

We have already published initial results generated by a prototype *hierarchical diff* [14]. Although we have an implementation roadmap for *hierarchical diff* based upon our prototypes, we have yet to implement the tool.

Our study will differ from other longitudinal studies of router configurations [4, 8, 11, 12] because of our ability to extract and compare blocks of code nested at arbitrary depth. Furthermore, our proposed tool is designed to be more general than other domain-specific diffing engines [10]. In addition, we will be experimenting with several different algorithms to combine differences in structure with differences in content [1, 3, 5].

3 Conclusion

If successful, system and network administrators will use our tools to extract and compare information from high-level configuration languages.

[1]This work was supported in part by the TCIPG project from DOE (under grant DE-OE0000097). Views are the authors' alone.

[2]We note that the term "lightweight" comes from Miller's Ph.D thesis [6].

References

[1] P. Bille. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 2005.

[2] Configuration management for agile system administrators - CFEngine. Retrieved November 11, 2011 from `http://cfengine.com`.

[3] S.S. Chawathe, A. Rajaraman, and J. Garcia-Molina, H. andn Widom. Change detection in hierarchically structured information. In *ACM SIGMOD International Conference on Management of Data*, pages 493–504. ACM, 1996.

[4] X. Chen, Z.M. Mao, and J. Van der Merwe. Towards automated network management: Network operations using dynamic views. In *The 2007 SIGCOMM Workshop on Internet Network Management*, pages 242–247. ACM, 2007.

[5] G. Cobena, S. Abiteboul, and A. Marian. Detecting changes in XML documents. *International Conference on Data Engineering*, 2002.

[6] R. Miller. *Lightweight Structure in Text*. PhD thesis, MIT, 2002.

[7] M. Patterson and Sassaman L. Exploiting the forest with trees. In *BlackHat*, 2010.

[8] D. Plonka and A.J. Tack. An analysis of network configuration artifacts. In *The 23rd Conference on Large Installation System Administration (LISA)*. USENIX, 2009.

[9] Puppet - overview - puppet labs. Retrieved November 11, 2011 from `http://projects.puppetlabs.com/projects/puppet`.

[10] R. Routray and S. Nadgowda. CIMDIFF: Advanced difference tracking tool for CIM compliant devices. In *Proceedings of the 23rd Conference on Large Installation System Administration*. USENIX Association, 2009.

[11] X. Sun, Y.W. Sung, S.D. Krothapalli, and S.G. Rao. A systematic approach for evolving VLAN designs. In *INFOCOM*, pages 1–9. IEEE, 2010.

[12] Y.W. Sung, S. Rao, S. Sen, and S. Leggett. Extracting network-wide correlated changes from longitudinal configuration data. In *10th Passive and Active Measurement Conference (PAM)*, 2009.

[13] A. Tsalolikhin. Configuration management summit. *USENIX login:*, 35:104–105, 2010.

[14] G. Weaver, N. Foti, S. Bratus, D. Rockmore, and S.W. Smith. Using Hierarchical Change Mining to Manage Network Security Policy Evolution. In *Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*. USENIX Association, 2011.