Jan Jürjens, Kurt Schneider (Hrsg.): Software Engineering 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 131

Understanding variable code: Reducing the complexity by integrating variability information

Dierk Lüdemann[1] Nazish Asad[2] Klaus Schmid[2] Christopher Voges[2]

Abstract: Um die Variabilität einer Software-Produktlinien zu handhaben, wird bei Verwendung einer C-basierten Sprache oftmals der integrierte Präprozessor genutzt. Eine bereits bewährte Methode [Sn96], um Zusammenhänge zwischen Präprozessorvariablen zu erkennen, ist der Einsatz der formalen Begriffsanalyse (FCA)[3]. Die aus der FCA resultierenden Verbände können jedoch sehr umfangreich sein. Daher schlagen wir ein neues Verfahren für Software-Produktlinien vor, mit dem die Größe der Verbände durch Berücksichtigung des Variabilitätsmodells reduziert werden kann. Mit Hilfe dieses neuen Verfahrens wird die FCA in unserer Arbeit zum ersten mal auf den Linux-Kernel angewendet. Dieser Beitrag erschien als Vollbeitrag bei der International Conference on Software Maintenance and Evolution (ICSME) 2016 [Lü16].[4]

Keywords: Reengineering, Produktlinien, Verband, Linux-Kernel, Variabilitätsmodell

1 Formale Begriffsanalyse für Software-Produktlinien

In C realisierte Software-Produktlinien setzen meist den C-Präprozessor zur Realisierung von Variabilität ein. Wenn in einer Implementierung viele Präprozessoranweisungen verwendet werden, dann sind deren Abhängigkeiten nur schwer nachvollziehbar. Ein Ansatz, um Zusammenhänge zwischen Präprozessorvariablen zu erkennen, ist die FCA [Sn96]. Dieser Ansatz repräsentiert den Zusammenhang zwischen Variabilität und Code als Begriffsverband. Die resultierenden Verbände können jedoch sehr komplex und umfangreich sein. So erhielten wir im Rahmen unserer Untersuchung von Linux Verbände mit bis zu 144 Begriffen. Daher bieten wir in diesem Beitrag ein neues Verfahren für Software-Produktlinien zur Reduktion der Verbände an. Dabei wird das Variabilitätsmodell der Software-Produktlinie extrahiert und verwendet, um Inkonsistenzen in den Verbänden zu finden. Der Verband wird dann entsprechend reduziert. Es handelt sich um die erste Anwendung der formalen Begriffsanalyse für Präprozessorvariablen auf eine der größten frei zugänglichen Software-Produktlinien, den Linux Kernel. Wir zeigen, dass unser Verfahren zu einer Reduktion der Begriffe des Begriffsverbandes führt und betrachten mit der architekturspezifischen Analyse eine spezielle Einsatzform des Verfahrens, bei der es zu besonders umfangreichen Reduktionen kommt. Durch diese Reduktionen ergibt sich eine Aufwandsreduktion für Folgeanalysen auf den Begriffsverbänden und somit eine einfachere Erkenntnisgewinnung der tatsächlichen Zusammenhänge von Präprozessoranweisungen und Variabilitäten im analysierten Quelltext.

[1] Universität Bremen, Bibliothekstraße 1, 28359 Bremen, dierk@tzi.de
[2] Universität Hildesheim, Universitätsplatz 1, 31141 Hildesheim, {asad, schmid, voges}@sse.uni-hildesheim.de

[3] Abgeleitet vom Englischen: Formal Concept Analysis

[4] Diese Arbeit wurde teilweise durch die Deutsche Forschungsgesellschaft im Projekt EvoLine als Teil des Schwerpunktprogramms SPP1593: Design for Future — Managed Software Evolution gefördert.

2 Analyseergebnisse für den Linux-Kernel und Schlussfolgerungen

In unserer Arbeit wird eine erweiterte FCA, die alle Konfigurationsmöglichkeiten des Linux-Kernels berücksichtigt, durchgeführt. Die Präprozessorbedingungen für die Begriffe werden sowohl für sich genommen auf Erfüllbarkeit hin untersucht als auch auf Konsistenz mit dem Variabilitätsmodell überprüft. Für den Linux-Kernel kann eine Reduktion durch die erweiterte FCA zwischen 1% und 3,5% der Begriffe erreicht werden für C-Dateien, die mehr als 20 verschiedene Präprozessorvariablen verwenden. Die Mehrheit der Reduktionen stammt jedoch von Inkonsistenzen innerhalb der Klauseln, die aus den verwendeten Präprozessoranweisungen entstehen und nicht von Verstößen gegen das Variabilitätsmodell. Somit kann unser Verfahren bereits ohne die Nutzung eines Variabilitätsmodells vorteilhaft eingesetzt werden. Die Analyse wurde für die Linux-Kernel Versionen 3.10.1 und 4.2.3 durchgeführt. Dabei erwiesen sich die Ergebnisse als stabil über die Versionen hinweg, da die Anzahl der Reduktionen fast gleichbleibend ist.

Die erweiterte FCA lässt sich auch explizit für eine einzige Hardware-Architektur durchführen.[5] In diesem Fall wird nur die Variabilität von einer Architektur wie *ARM*, *x86* oder *PowerPC* berücksichtigt. Das führt zu wesentlich mehr Einschränkungen für die erlaubten Konfigurationsmöglichkeiten des Linux-Kernels. Unter diesen Bedingungen lassen sich zwischen 10% und 30% der Begriffe reduzieren, abhängig von der betrachteten Architektur. Dies ist eine wesentliche Verbesserung für alle möglichen nachfolgenden Analysen basierend auf dem Begriffsverband. Das gilt insbesondere, wenn dieser Verband manuell navigiert und betrachtet wird. Es zeigt den potentiellen Einfluss unseres erweiterten FCA-Verfahrens für die Untersuchung der Zusammenhänge zwischen Präprozessorvariablen im Kontext eingeschränkter Variabilität.

Eine Analyse des kompletten Linux Kernels benötigt einige Stunden auf Grund der Komplexität des Variabilitätsmodells von Linux und der Anzahl an Quelltextdateien. Jedoch lässt sich die erweiterte FCA auch für die kompliziertesten C-Dateien des Linux-Kernels innerhalb einer Minute durchführen, sofern nur eine konkrete Linux-Architektur zu berücksichtigen ist. Der Grund für den hohen Gesamtaufwand liegt in der hohen Zahl von Dateien in der Linux-Implementierung.

Literaturverzeichnis

[Lü16] Lüdemann, Dierk; Asad, Nazish; Schmid, Klaus; Voges, Christopher: Understanding Variable Code: Reducing the Complexity by Integrating Variability Information. In: 32nd IEEE International Conference on Software Maintenance and Evolution. S. 312–322, 2016.

[Sn96] Snelting, Gregor: Reengineering of Configurations Based on Mathematical Concept Analysis. ACM Transactions on Software Engineering and Methodology (TOSEM), 5(2):146–189, 1996.

[5] Die Implementierung von Linux behandelt die Konfigurationsmöglichkeiten der unterschiedlichen Hardwareplattformen als getrennte Variabilitätsmodelle.