Zertifizierung einer Sicherungskomponente mittels durchgängig formaler Modellierung*

Udo Frese[1], Daniel Hausmann[2], Christoph Lüth[1], Holger Täubig[1], Dennis Walter[1]

[1] Deutsches Forschungszentrum für Künstliche Intelligenz, Bremen
[2] FB 3 — Mathematik und Informatik, Universität Bremen
christoph.lueth@dfki.de

1 Einführung

Dieses Papier berichtet über Arbeiten im Rahmen des Projektes SAMS, in welchem eine Fahrwegsicherungskomponente für Serviceroboter und fahrerlose Transportsysteme entwickelt und ihre Software nach IEC 61508 zertifiziert wird. Neu ist hierbei der durchgehende Einsatz formaler Modellierung und Beweis als Mittel zur Zertifizierung, das es uns erlaubt, die erforderlichen Sicherheitsbedingungen sehr abstrakt mathematisch zu formulieren, und gleichzeitig den Bezug zur Implementierung bewiesen korrekt herzustellen.

Indem wir die Grundlagen der Anwendungsdomäne (Geometrie und die Mechanik bewegter Objekte) in einem Theorembeweiser formalisieren, können wir die Sicherheitsanforderungen auf einer abstrakten, mathematischen Ebene formulieren. Diese sind dann wesentlich leichter zu validieren als implementationsnah formulierte Programmeigenschaften. Darüber hinaus modellieren wir die Implementierung innerhalb desselben formalen Rahmens. Dadurch erhalten wir einen nahtlosen Entwicklungsprozess mit einem klaren und beweisbar korrekten Übergang von den Sicherheitseigenschaften zu der Implementation.

Wir glauben, dass diese Vorgehensweise für alle Sicherheitszertifikationen nützlich sein kann: die formale Modellierung erzwingt, dass alle Annahmen, die in den Sicherheitsanforderungen, der Implementation oder den Beweisen implizit enthalten sind, explizit gemacht werden, so dass sich der Zertifikationsprozess auf die Sicherheitsanforderungen konzentrieren kann, da die Korrektheitsbeweise automatisch geprüft werden können.

2 Das Projekt SAMS

Ziel des Projektes SAMS ist die Entwicklung einer Fahrwegsicherungskomponente für Serviceroboter und fahrerlose Transportsysteme, welche die Zulassungsvoraussetzungen

*Forschung gefördert durch das BMBF im Rahmen der *Leitinnovation Servicerobotik*, Fkz. 01 IM F02

für Sicherheitsgeräte entsprechend IEC 61508 erfüllt. Die Komponente überwacht mittels eines Sicherheitslaserscanners ein Schutzfeld, das sich dem veränderlichen Zustand (Geschwindigkeit, Richtung, Lenkwinkel etc.) des Fahrzeugs dynamisch anpasst, und veranlasst rechtzeitig vor der Kollision mit einem Hindernis eine Bremsung. Industriell verfügbare Lösungen arbeiten nur mit wenigen vorkonfigurierten Schutzfeldern, was Bahnverlauf und Geschwindigkeit unnötig einschränkt

Für Forschungsprototypen ist die dynamische Schutzfeldberechnung Stand der Technik, aber keine existierende Implementation ist nach den entsprechend der Maschinenrichtlinie erforderlichen Normen zertifiziert worden. Kernvorhaben des Projektes ist deshalb die Zertifizierung einer dynamischen Schutzfeldberechnung nach der relevanten Norm IEC 61508 als eine Sicherheitskomponente, die für Anwendungen bis SIL-3 geeignet ist. Dieses ist sowohl für die Robotik als auch für die Sicherheitszertifikation neu, da typischerweise die Software zertifizierter Sicherheitseinrichtungen algorithmisch sehr einfach ist und Normen wie DIN EN ISO 10218 hardwarebasierten Lösungen wie Lichtvorhängen oder Käfige betonen.

3 Sicherheitsanforderungen und ihr Nachweis

Die globale Sicherheitsanforderung des zu entwickelnden Systems ist auf informelle Weise sehr einfach zu formulieren. Es muss garantiert werden, dass das gesicherte Gerät (der Roboter) nicht mit Hindernissen —insbesondere in der Nähe stehenden Personen— kollidiert. Die durch die Sicherungskomponente erhöhte Flexibilität im Fahrverhalten geht mit einer erhöhten Komplexität der eingesetzten Software einher. Hieraus ergeben sich weitere Sicherheitsanforderungen auf Modulebene, welche in ihrer Gesamtheit die Erfüllung der globalen Sicherheitsanforderung ergeben müssen.

In informellen Verifikationsansätzen kann der Schluss von der Erfülltheit der tiefer liegenden Anforderungen auf die Erfülltheit der globalen Anforderungen selbstverständlich auch nur informell gezogen werden, so dass z.B. Anforderung übersehen oder Annahmen nur implizit gemacht werden (z. B. dass mithilfe des Laserscanners auch wirklich alle Hindernisse erkannt werden können). Demgegenüber werden in der durchgängig formalen Modellierung und Verifikation —ausgehend von Eigenschaften und Gestalt des Roboters und der Umgebung, über die Sicherheitsanforderungen, bis hin zum Quellcode— derartige Fehler vermeiden, und damit trotz hoher Komplexität des entwickelten Systems Vertrauen in dessen Korrektheit geschafft. Als Modellierungs- und Beweiswerkzeug kommt in diesem Projekt Isabelle/HOL zum Einsatz, mit dem wir gute Erfahrungen gesammelt haben.

Domänenmodellierung. Die Welt, d. h. der Roboter und seine Umgebung, werden in der reellen Ebene modelliert. Dies ist adäquat, da die Sensoren des Roboters (Odometrie und Laserscanner) nur zweidimensionale Informationen liefern, und der Roboter ausschließlich auf planen Flächen zum Einsatz kommt.

Da sich die Objekte einschließlich des Roboters bewegen oder ihre Gestalt verändern, werden sie als über der Zeit veränderliche Funktionen des Typs $\mathbb{R}_0^+ \to \alpha$ dargestellt (wobei

```
/*@ requires \valid_ptr(v, 0, len)
  @ ensures (\forall int i; 0 <= i &&
  @                  i < len --> v[i] <= \result) &&
  @          (glob == \old(glob) + 1)  */
  int f(int *v, int len);
```

Abbildung 1: Einfaches Beispiel für die Spezifikation einer Funktion `f`; der Rückgabewert wird als größer als die Elemente des Feldes `v` spezifiziert.

$\mathbb{R}_0^+$ als Zeit interpretiert wird, und α ein beliebiger Typ ist.) Wir nennen solche Funktionen *Verhalten*. So liefert etwa die Funktion $robot : \mathbb{R}_0^+ \to \mathbb{R}^2$ die Position des Roboters zu jedem Zeitpunkt. Eigenschaften wie die Stetigkeit von Funktionen lassen sich in Isabelle/HOL einfach spezifizieren, und die globale Sicherheitsanforderung läßt sich leicht vereinfacht wie folgt formalisieren:

$$[\forall t,\, t'.\, \forall o \in Obs.\, o(t) = o(t')] \Longrightarrow \forall t.\, \forall o \in Obs.\, robot(t) \notin o(t) \tag{1}$$

wobei Obs die Menge aller Funktionen $\mathbb{R}_0^+ \to \mathcal{P}(\mathbb{R}^2)$ ist, die Objekte repräsentieren.

Einbettung des Kontrollprogramms in das Weltmodell. Um nachzuweisen, dass *robot* die Eigenschaft (1) erfüllt, müssen wir die Tatsache ausdrücken, dass der Roboter durch sein Kontrollprogramm gesteuert wird, welches in unserem Fall ein reaktives System ist. Dies bedeutet, dass die eigentliche Sicherheitsroutine alle N Millisekunden mit den aktuell gelesenen Sensorwerten aufgerufen wird, und einen Rückgabewert berechnet, der angibt, ob ein Nothalt eingeleitet werden muss. Die Zykluslänge wird durch die Scanfrequenz des Laserscanners bestimmt. Hierfür werden *diskrete Verhaltensfunktionen* des Typs $\mathbb{N} \to \alpha$ eingeführt, wobei jede natürliche Zahl genau einen Zyklus repräsentiert. Diskrete Verhalten sind das Verbindungsglied zwischen Weltmodell und konkretem Sicherungsprogramm, anhand derer wir spezifizieren können, welchen Einfluss das Sicherungsprogramm auf die reale Welt hat. Insbesondere das diskrete Verhalten $stop$ nimmt Einfluss auf das Roboterverhalten $robot$: ab dem Zyklus, in dem ein Nothalt emittiert wird ($stop(n) = True$), bremst der Roboter bis zum Stillstand mit definierter Verzögerung ab.

Am unteren Ende der Abstraktionshierarchie im Entwicklungsprozess steht der konkrete Code, welcher den Sicherungsalgorithmus implementiert. Dieser ist im vorliegenden Projekt in C nach den MISRA Richtlinien [MIS04] geschrieben, und hat im Wesentlichen sicherzustellen, dass der Roboter darf seine Fahrt nur genau dann fortsetzt, wenn anhand der verfügbaren Sensordaten sichergestellt werden kann, dass innerhalb des nächsten Zyklus keine Notbremsung eingeleitet werden muss. Um Eigenschaften von C-Programmen in unserem Rahmenwerk zu formulieren und zu beweisen, betten wir wie JML [BCC+05] oder Caduceus [FM04] Spezifikationen in den Quellcode ein; Abb. 1 zeigt ein Beispiel.

Die formale Verbindung zwischen dem Ausgabeverhalten *stop* und dem C-Sicherungsprogramm wird hergestellt, indem wir die Semantik von C sowie der eingebetteten Spezifikationssprache in Isabelle/HOL definieren. Die Semantik eines C-Programms ist dann ein Zustandstransformer, d. h. eine Funktion, die einen Eingabezustand erwartet —dieser

beinhaltet die Programmvariablen sowie aktuelle Sensordaten— und einen Ausgabezustand sowie ein Resultat liefert. Ein Zustandstransformer lässt sich leicht in ein diskretes Verhalten übersetzen: in jedem Zyklus wird der Transformer mit dem Zustand des letzten Durchlaufs aufgerufen, wobei dieser Vorzustand noch um die aktuellen Sensordaten ergänzt wird. Damit ist die Einbindung des C-Sicherungsprogramms und seiner Eigenschaften in das Weltmodell gelungen: die Semantik des Programms ist Teil der Definition des Nothalt-Verhaltens $stop$.

4 Schlussbemerkungen

In unser Herangehensweise wird die gesamte Entwicklung, von den Sicherheitsanforderungen bis hin zur Implementation, innerhalb von Isabelle/HOL formalisiert. Die Vorteile dieser Methodologie sind zum ersten, dass die Spezifikation sehr abstrakt formuliert werden kann, mit der Betonung auf den Konzepten der Domänenmodellierung und nicht der Implementation, so dass die Zertifikation auf dieser Ebene die Validität der Sicherheitsanforderung diskutieren und früh eventuelle konzeptionelle Fehler offenlegen kann; zum zweiten können Beweise maschinell geprüft werden, so dass sich die Zertifikation auf diese Kernfragestellungen konzentrieren kann; und zum dritten werden durch den formalen Beweis von Eigenschaften oft versteckte Annahmen in Spezifikation und Implementation aufgedeckt. Es kann also formal bewiesen werden, dass die Implementierung wirklich für eine korrekte Ausführung der Sicherheitsfunktion Kollisionsvermeidung sorgt. Alle Annahmen über das Roboterverhalten, die Umgebung, oder auch das Ausführungsmodell des C-Codes sind explizit in Isabelle/HOL formuliert. Fehlende Annahmen werden offengelegt, wenn Beweise nicht vollendet werden können.

Ein Nachteil dieser Herangehensweise ist, dass formale Beweise vergleichsweise aufwändig sind. Das kann vielleicht dadurch aufgewogen werden, dass einmal bewiesene Theoreme für Projekte mit der gleichen Anwendungsdomäne wiederverwendet werden können. Ferner ist die Akzeptanz durch die Zertifizierungsbehörden noch nicht gegeben. Hier ist zu hoffen, dass das SAMS-Projekt, in dem die Zertifizierung durch den TÜV Süd durchgeführt wird, Brücken schlagen kann, denn letztendlich können alle Seiten (Zertifizierer, Implementierer und sogar Endnutzer) von den Vorteilen profitieren.

Literatur

[BCC+05] Lilian Burdy, Yoonsik Cheon, David Cok, Michael Ernst, Joe Kiniry, Gary T. Leavens, K. Rustan M. Leino und Erik Poll. An overview of JML tools and applications. *International Journal on Software Tools for Technology Transfer*, 7(3):212– 232, June 2005.

[FM04] Jean-Christophe Filliâtre und Claude Marché. Multi-Prover Verification of C Programs. In *Sixth International Conference on Formal Engineering Methods (ICFEM)*, Jgg. 3308 of *Lecture Notes in Computer Science*, Seiten 15–29, Seattle, November 2004. Springer.

[MIS04] MISRA-C: 2004. Guidelines for the use of the C language in critical systems., 2004.