Sicherheit in der Künstlichen Intelligenz

Christoph Lüth, Bernd Krieg-Brückner

Dieses Jahr wurde die Künstliche Intelligenz fünfzig Jahre alt. Aus einem esoterischen Randbereich der Informatik ist eine Kerndisziplin geworden. Techniken der Künstlichen Intelligenz (KI) werden inzwischen in fast allen Gebieten der Informatik eingesetzt. In dem Maße, in dem sich diese Techniken verbreiten, stellt sich verstärkt die Frage nach der Sicherheit. Wenn ich künftig einen Serviceroboter einsetze, wie kann ich sicher sein, dass er mich nicht verletzt? In diesem Artikel stellen wir drei Techniken vor, mit denen das Vertrauen in die Zuverlässigkeit von KI-Systemen erhöht werden kann. Als Beispiele betrachten wir typische KI-Systeme wie Theorembeweiser und Roboter. Für beide ist Sicherheit unerlässlich: ein Theorembeweiser ist nur dann nützlich, wenn wir sicher sein können, dass er nur korrekte Beweise erzeugt (und nicht etwa Widersprüche), und ein mobiler Roboter sollte nicht mit Hindernissen kollidieren.

1 Einleitung

Zur der Frage nach der Sicherheit in der KI gilt es erst einmal den Begriff der *Sicherheit* zu klären, unter dem im Deutschen zwei im internationalen Sprachgebrauch getrennte Aspekte zusammengefasst werden:

- Unter *safety* verstehen wir Funktionssicherheit, insbesondere in Bezug auf Gefahr für Leib und Leben, und
- unter *security* verstehen wir Daten- oder Informationssicherheit, z.B. Integrität und Schutz gegen unerlaubten Zugriff.

Der Schwerpunkt dieses Beitrages wird der erste Aspekt sein, insbesondere wie wir die *Korrektheit* von Systemen sicherstellen können, die Techniken der KI nutzen.

Ferner gilt es festzustellen, was immer wieder betont werden muss: Sicherheit ist eine *durchgängige Eigenschaft*. Mit anderen Worten, Sicherheit ist ein Aspekt, der vom Entwurf bis zu Implementierung und Test in allen Phasen der Entwicklung berücksichtigt werden muss. Das ist natürlich keine Besonderheit der Künstlichen Intelligenz, gerät aber bei anspruchsvollen Algorithmen und Systemarchitekturen, wie sie in der künstlichen Intelligenz verwendet werden, leichter in Vergessenheit.

2 Sicherheit durch Konstruktion

Die Korrektheit eines Systems kann schon bei dessen Konstruktion leichter sichergestellt werden, wenn die Korrektheit (und damit Sicherheit) des Gesamtsystems auf die Korrektheit möglichst weniger Teile reduziert wird, indem sicherheitskritische Funktionalität in möglichst wenigen (und kleinen) Modulen implementiert und dort verkapselt wird. Dies ist nichts weiter als gute Praxis der Softwareentwicklung.

Anstatt bei einem Theorembeweiser eine große Anzahl von Beweisprozeduren, Heuristiken und Entscheidungsprozeduren zu implementieren, die alle auf einer konkreten Repräsentation einer Termsprache rechnen, wird ein kleiner logischer Kern definiert, der einen abstrakten Datentyp für Theoreme implementiert, und der nur Funktionen auf diesem Datentyp bereitstellt, die korrekten Ableitungen entsprechen (beispielsweise Instantiierung von Variablen in einem Theorem, oder eine Schnittregel). Auf diese Weise reduziert sich beispielsweise die Korrektheit des Beweisers Isabelle [5] bei einem Gesamtumfang von ca. 150.000 Zeilen Programmcode und 300.000 Zeilen Beweisskripten auf einen logischen Kern von ca. 3.000 Zeilen.

Bei einem Roboter kann die Sicherheit garantiert werden, indem der Zugriff auf die Aktuatoren auf ein Sicherheitsmodul begrenzt wird, welches mit geeigneter Sensorik sicherstellt, dass nur jene Steuerungskommandos ausgeführt werden, die die Sicherheit des Roboters nicht gefährden. Module in höheren Schichten müssen nicht sicher sein, da ein eventuell unsicheres Steuerungskommando von dem Sicherheitsmodul abgefangen wird. Der autonome Bremer Rollstuhl *Rolland* [4] verfügt beispielsweise über ein Sicherheitsmodul, das mit Hilfe eines Sicherheitslaserscanners eine lokale Karte der Umgebung erstellt und jeden Steuerungsbefehl daraufhin überprüft, ob er zu einem Kurs führt, bei dem vor einem Hindernis nicht mehr gebremst werden kann. Ein über dieser Sicherheitschicht liegendes Modul plant Umgehungen um Hindernisse, was aber jetzt nicht sicherheitskritisch ist, da das Sicherheitsmodul die Kollisionsfreiheit sicherstellt. Damit wird das Problem der Sicherheit auf Verfügbarkeit reduziert: das System ist auf jeden Fall sicher, aber eventuell nicht immer verfügbar.

3 Sicherheit durch Überprüfung

Eine andere Möglichkeit, die Korrektheit einer Berechnung sicherzustellen, ist die Korrektheit einer (wesentlich einfacheren) Funktion sicherzustellen, die das Ergebnis der eigentlichen Berechnung prüft. Diese Vorgehensweise eignet sich dann besonders gut, wenn die Überprüfung wesentlich einfacher ist als die Konstruktion; zum Beispiel ist es einfacher, zu prüfen ob eine Zahl eine Nullstelle eines Polynoms ist, als die Nullstellen zu berechnen.

In dem Fall eines Roboters wie eines Manipulatorarms kann es einfacher sein, für einen Bahnplaner eine kleine Zusatzfunktion zu implementieren und zu verifizieren, die prüft, ob eine konstruierte Bahn sicher ist, anstatt zu versuchen, von vornherein nur sichere Bahnen zu konstruieren.

Theorembeweiser, die nach diesem Prinzip arbeiten, erzeugen mit der Herleitung eines neuen Theorems ein Beweisobjekt, das einfach auf Korrektheit geprüft werden kann. Oft werden Beweise mit dem Curry-Howard-Isomorphismus in Terme des getypten Lambda-Kalküls übersetzt; der Beweis ist korrekt genau dann, wenn der angegeben Typ korrekt ist. Damit wird die Frage der Korrektheit des Beweises auf eine Typüberprüfung im getypten Lambda-Kalkül reduziert, die schnell durch statische Analyse erfolgen kann.

4 Sicherheit durch formalen Beweis

Die oben vorgestellten Techniken dienen der Reduzierung der Frage der Korrektheit des Gesamtsystems auf die Korrektheit kleinerer Teile. Es bleibt die Frage, wie deren Korrektheit gezeigt werden kann. Jeder Informatiker hat im Laufe seines Studiums einmal eine Funktion im Hoare-Kalkül verifiziert, und musste die Erfahrung machen, dass dieser Beweis länger und komplizierter ist als die Funktion selbst. Wie können wir solchen Beweisen trauen? Indem wir sie von einer Maschine prüfen lassen.

In der Hardwareindustrie, die mit weit geringeren Fehlertoleranzen und höheren Fehlerfolgekosten arbeitet als die Softwareindustrie, ist die formale Verifikation kritischer Teile komplexer Chips inzwischen Stand der Technik (zum Beispiel Fließkommaarithmetik in den Pentium-Prozessoren der Firma Intel [2]). In der Softwareindustrie ist dies trotz großer Fortschritte (wie das Projekt Verisoft[1]) immer noch die große Ausnahme, aber es gibt Hinweise darauf, dass sich dieses in den nächsten fünf Jahren ändern wird (unter anderem durch die deutlich gestiegene Leistungsfähigkeit halbautomatischer Theorembeweiser wie Isabelle, PVS oder Coq). Dabei soll nicht verhehlt werden, dass formales Beweisen mit diesen Werkzeugen immer noch extrem arbeitsaufwändig, und daher nur bei hohen Sicherheitsanforderungen gerechtfertigt ist.

5 Offene Fragen

Leider sind die oben angeführten Techniken nicht in allen Situationen gleich hilfreich. Einige Algorithmen und Systeme entziehen sich schon von ihrer Konstruktion her einer derartigen Sicherheitsbetrachtung; hier ist eine separate Sicherheitsbetrachtung nötig.

Ein Beispiel sind probabilistische Algorithmen, wie zum Beispiel die Markov-Lokalisation. Diese macht Aussagen darüber, wo ein Roboter sich am wahrscheinlichsten befindet. Sicherheitsgerichtet ist es allerdings weniger relevant, wo (genau) sich der Roboter befindet, sondern es muss ausgeschlossen werden, dass der Roboter sich an Orten befindet, die als gefährlich gelten und nicht mit anderer Sensorik erkannt werden können (zum Beispiel am Kopf einer Treppe). Hier muss ein grundlegend anderer Ansatz gewählt werden, der von sehr konservativen Annahmen über Karte und Umgebung ausgehend eine zwar ungenauere, aber dafür sicherheitsgerichtete Lokalisation ermöglicht. Dies kann dazu führen, dass in einem autonomen System zwei Lokalisationsalgorithmen verwendet werden müssen: einer im Sicherheitsmodul für eine sicherheitsgerichtete Lokalisation, und ein anderer in der (nicht sicherheitsgerichteten) Bahnplanung für optimale Lokalisation.

Ein weiteres Beispiel sind selbstlernende Algorithmen, wie sie zum Beispiel im achtbeinigen Laufroboter SCORPION [3] Verwendung finden. Die Bewegung der Beine des SCORPION ist nach einem realen Skorpion modelliert und selbstlernend; es gibt kein zentrales Steuerprogramm. Hier müsste ein Sicherheitsmodul implementiert werden, das die Bewegung überwacht und im Notfall eingreift.

6 Zusammenfassung

Sicherheit ist ein Thema, dem in der Künstlichen Intelligenz bis jetzt noch zu wenig Achtung geschenkt wurde, insbesondere weil Sicherheit eine durchgängige Eigenschaft ist, die schon beim Entwurf mit bedacht werden muß. Wir haben Techniken vorgestellt, die die Sicherheit verbessern können, aber in komplexen Anwendungsdomänen wie der Robotik sind noch verstärkte Forschungsanstrengungen nötig.

Literatur

[1] The Verisoft project. `http://www.verisoft.de`.

[2] R. Kaivola. Formal verification of Pentium 4 components with symbolic simulation and inductive invariants. In *CAV'2005*, Springer LNCS 3576, S. 170– 184, 2005.

[3] F. Kirchner, D. Spenneberg, and R. Linnemann. A biologically inspired approach towards robust real world locomotion in an 8-legged robot. In J. Ayers *et al* (Hrsg.), *Neurotechnology for Biomimetic Robots*. MIT-Press, 2002.

[4] A. Lankenau and T. Röfer. A safe and versatile mobility assistant. *Reinventing the Wheelchair. IEEE Robotics and Automation Magazine*, 8(1):29 – 37, 2001.

[5] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, Springer LNCS 2283, 2002.

Kontakt
Christoph Lüth Ph.D., Prof. Dr. Bernd Krieg-Brückner
DFKI GmbH, Sichere Kognitive Systeme
Enrique-Schmidt-Straße 5, 28359 Bremen
Email: sks@dfki.de

Christoph Lüth ist stellvertretender Leiter des Bereiches *Sichere Kognitive Systeme* des DFKI-Labors Bremen. Nach seiner Promotion an der University of Edinburgh arbeitete er als Wissenschaftlicher Assistent an der Universität Bremen. Seine Forschungsschwerpunkte sind formale Programmentwicklung und Verifikation, insbesondere im Bereich der Robotik.

Bernd Krieg-Brückner ist seit 1982 Professor an der Universität Bremen. Neben dem Gebiet der Formalen Methoden und Werkzeuge zur Entwicklung korrekter Software baute er das Gebiet der Kognitiven Robotik auf, ist am SFB/TR8 *Spatial Cognition* beteiligt und leitet seit Ende 2005 den Bereich *Sichere Kognitive Systeme* des DFKI-Labors Bremen.

Auszug aus: Künstliche Intelligenz, Heft 1/2007, ISSN 0933-1875, BöttcherIT Verlag, Bremen, **www.kuenstliche-intelligenz.de/order**