University of Applied Sciences

HOCHSCHULE
EMDEN·LEER

Definition und Implementierung einer formalen Sprache zur Spezifikation, Verifikation und Implementierung von parallelen und verteilten Systemen

Gert Veltink

6 May, 2015

HS Emden/Leer T 49 4921 807.1803 (P2) gert.veltink@hs-emden-leer.de

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Zusammenfassung

Einführung

Die (Computer)Welt wird sich zukünftig immer mehr mit parallelen und verteilten Systemen auseinandersetzen (müssen)!

Die Gründe hierfür liegen vor allem in der Tatsache, dass die physikalischen Grenzen der Rechner-Hardware mittlerweile erreicht oder fast erreicht worden sind. War vor etwa fünfzehn Jahren noch eine deutliche Tendenz zu erkennen, dass Prozessoren mit immer höheren Taktfrequenzen produziert wurden, so hat sich in den letzten Jahren abgezeichnet, dass die Produzenten von Computer-Hardware immer häufiger die letzte Möglichkeit der Skalierung der Fähigkeiten ihrer Produkte versuchen auszunutzen, nämlich indem Berechnungen gleichzeitig (parallel) ausgeführt werden. Dies hat dazu geführt, dass heutzutage sogar mobile Endgeräte, wie z.B. Smartphones, Mehrkernprozessoren enthalten.

Gerade die starke Entwicklung der mobilen Endgeräte zeigt, dass die meisten Anwendungen nicht länger nur 'lokal' arbeiten, sondern im ständigen Kontakt mit anderen Systemen im Internet stehen, um Ihre Aufgaben erfüllen zu können. Die Bearbeitung der Daten findet deswegen nicht mehr in einem lokalen System, sondern per Definition in einem verteilten System statt.

Diese Entwicklung zur Parallelisierung und Verteilung der Arbeit ist auch sehr deutlich in der Industrie, insbesondere in der Automatisierungstechnik, erkennbar und wird hier momentan u.a. unter dem Begriff '*Industrie 4.0*' gehandelt.

Das Ziel dieses Forschungsvorhabens ist, ein formales Framework für die Spezifikation, Simulation, Verifikation und Implementierung von parallelen und verteilten Prozessen zu bieten mit einer Fokussierung auf die Aufgaben in der Automatisierungstechnik, damit die Resultate direkt in der digitalen Fabrik der Abteilung E+I eingesetzt werden können und als Basis für weitere Entwicklungen in diesem Bereich dienen können.

Formale Methoden

Damit eine Verifikation der Spezifikationen möglich ist, muss die Sprache auf formalen, mathematischen Prinzipien basieren. In diesem Projekt wird deswegen eine Prozessalgebra mit einer axiomatischen Basis genutzt.

Als Ausgangspunkt dient PSF, eine formale Beschreibungssprache für die Spezifikation und Analyse von parallelen kommunizierenden Systemen. Die Sprache integriert in sich eine Prozessalgebra für die Spezifikation von Prozess-Verhalten, eine Sprache für die algebraische Spezifikation von abstrakten Datentypen und eine Implementierung der Modul-Algebra, um die Strukturierung von größeren Spezifikationen zu unterstützen. Der Prozessteil in PSF ist eine rechnerlesbare Variante von ACP (Algebra of Communicating Processes). ACP wurde entwickelt von Bergstra & Klop [BK85] und gehört zu der Familie der Prozesskalküle, wie z.B. auch CSP (Communicating Sequential Processes) von Hoare [Hoa78] und CCS (Calculus of Communicating Systems) von Milner [Mil80]. Die Beschreibung der abstrakten Datentypen sowie die Modularisierungsprinzipien in PSF wurden aus ASF (Algebraic Specification Formalism) von Heering & Klint [BHK89] übernommen.

Im vorliegenden Vorschlag wird, anders als in PSF, als Basis der Prozess-Spezifikation TCPτ [BBR10] verwendet, eine relativ neue Prozessalgebra die Anteile aus CCS, CSP und ACP in sich vereint.

Das übergeordnete Ziel der Prozessalgebren ist, auch nach mehr als zwanzig Jahren, noch immer eine Unterstützung zu bieten bei der formalen Konstruktion von Rechneranwendungen. Der große Vorteil dieser Formalismen ist, dass sie 'mathematisch korrekt' sind. Ein wichtiger Nachteil für viele potentielle Anwender ist jedoch, dass diese Formalismen eben '(zu) mathematisch' sind.
Ein weiteres Ziel des Forschungsvorhabens ist dann auch, die Sprache so umzugestalten, dass sie intuitiv eher als eine Erweiterung einer Programmiersprache als eine formale Spezifikationssprache verstanden werden kann. Damit wird hoffentlich erreicht, dass die Hemmschwelle zur Benutzung derartiger Formalismen gesenkt werden kann.

Ziele

Das essentielle Ziel dieses Forschungsvorhabens ist es, eine neue formale ereignisbasierte Sprache zu definieren für das Beschreiben von Prozess-Workflows und evtl. von 'Orchestration' im industriellen Automatisierungs-Umfeld. Obwohl die Sprache prinzipiell auch als eine Alternative für Formalismen, wie definiert in IEC 61131-3, verstanden werden kann, soll die Sprache in erster Linie benutzt werden, um die 'externen' Schnittstellen von bereits existierenden Systemen zu definieren. Anschliessend folgt mit der Definition der Kommunikationsmöglichkeiten zwischen diesen Teilsystemen und mittels der Kompositionalitäts-Prinzipien der Prozessalgebra die Definition des Gesamtverhalten des Zusammenschlusses der bereits existierenden Systeme.

Die Definition und vor allem die Implementierung einer neuen Programmiersprache erfordert sehr viel Arbeit. Eine Möglichkeit den Implementierungsaufwand zu reduzieren, ist die Wiederverwendung von bereits existierenden Sprachen, die nur in bestimmten Bereichen erweitert werden. Eine besonders elegante Form einer Erweiterung bieten Sprachen, die die Erstellung von sogenannten 'Domain Specific Languages' (DSL) unterstützen. Ein Beispiel einer solchen Sprache ist Scala, ein weiteres Beispiel ist Xtend. Beide Sprachen sind Weiterentwicklungen von Java und basieren auf der JVM (Java Virtual Machine), so dass viele Java-Prinzipien in der Syntax erkennbar sind und auch die existierenden Java-Bibliotheken wiederverwendet werden können.

Anders als in PSF ist in diesem Forschungsvorhaben vorgesehen, dass die Datentypen und Modularisierung-Konstrukte der Wirtssprache benutzt werden. Auch dies senkt den Aufwand der Implementierung und erhöht gleichzeitig den Wiedererkennungswert für Entwickler, die bereits Erfahrung mit Java oder ähnlichen Programmiersprachen haben. *Appendix B* zeigt einen Teil einer Spezifikation als Beispiel für die Syntax der neuen Sprache. Als Kontrastprogramm zeigt *Appendix A* den gleichen Ausschnitt dieser Spezifikation in TCPτ.

Bisherige Arbeiten

Dieses Forschungsvorhaben schliesst an die Ergebnisse des Forschungssemesters im Sommersemester 2010 an. Weil damals die benötigte Hardware leider erst ein Monat vor Ende des Semesters zur Verfügung gestellt werden konnte, war die praktische Forschung und Realisierung in 2010 nicht möglich. Die Zeit wurde jedoch verwendet zur Erstellung eines Papers, das letztendlich in *Fundamenta Informaticae* publiziert wurde [Vel10].

In diesem Dokument wurde bereits die damals gerade publizierte Zusammenführung der drei wichtigsten Prozessalgebren (CCS, CSP, ACP) in TCPτ angesprochen sowie die Tatsache, dass die Arbeit an einem einfachen TCPτ-Simulator für Unterrichtszwecke auf Java-Basis angefangen wurde.

Aus diesem Java-Simulator wurde später, in einer Bachelorarbeit von Andreas Breer, ein Simulator in Scala, der durch Unterstützung von domänenspezifischen Sprachelementen in Scala eine natürlichere Spezifikationssprache ermöglicht. Die Ergebnisse der Bachelorarbeit wurden erweitert und dies resultierte in einer Publikation und Präsentation auf der EKA 2014 in Magdeburg [BV14].

Im Moment untersucht Benoit Verhaeghe (ein französischer Erasmus-Student) die Möglichkeit die Java Virtual Machine zu verbinden mit der OPC Unified Architecture. Die OPC/UA ist ein industrieller Standard für Maschine-Maschine-Kommunikation (M2M), die in der digitalen Fabrik der Abteilung E+I bereits eingesetzt wird. Mittels der JVM-OPC/UA Schnittstelle wird es zukünftig möglich sein, Spezifikationen in TCPτ nicht nur zu simulieren, sondern auch wirklich auszuführen, z.B. in der digitalen Fabrik.

Verbindung von Forschung und Lehre

Das Forschungsvorhaben passt direkt zum Forschungsschwerpunkt 'Industrial Informatics' der Hochschule Emden/Leer. Es schliesst außerdem direkt an bei der Lehre, insbesondere bei der Veranstaltung "Parallele und Verteilte Systeme" aus dem Angebot des Studiengangs Master-Medieninformatik, die ich persönlich für Studierenden des gesamten VFH-Verbundes anbiete sowie bei der Veranstaltung "Formale Methoden zur Entwicklung intelligenter Produktionssysteme" vom Kollegen Colombo, in der ich jährlich auch einige Vorlesungen übernehme. In beiden Veranstaltungen wird die bereits existierende Scala-Implementierung des TCPτ-Simulators für praktische Übungen eingesetzt.

Ziel ist es, die derzeit gehandhabte Sprache so weit zu entwickeln, dass sie nicht nur für reine Forschungsarbeiten, sondern auch für Projekte im industriellen Umfeld und in Zusammenarbeit mit der Industrie eingesetzt werden kann. Damit entsteht dann automatisch die Möglichkeit nachfolgende Projektarbeiten für Studierenden zu definieren, um zum einen Aspekte der digitalen Fabrik formal zu spezifizieren und zu verifizieren und zum anderen dabei wichtige Rückmeldungen zu gewinnen für die Weiterentwicklung der Sprache und der zugehörigen Werkzeuge.

Appendix A

Es folgt ein Ausschnitt einer Spezifikation in der kompakten mathematischen Darstellung TCPτ. (Quelle: [BBR10])

$$
\begin{aligned}
S &= 1 + S0 \cdot S1 \cdot S, \\
Sn &= 1 + \sum_{d \in D} i?d.Sn_d, \\
Sn_d &= sk!dn.Tn_d, \\
Tn_d &= ls?(1-n).Sn_d + ls?\bot.Sn_d + ls?n.1,
\end{aligned}
$$

Appendix B

Es folgt ein Ausschnitt einer Spezifikation, der die bis jetzt angedachte neue Syntax der Sprache zeigt.

```
/*
 * This specification defines the Alternating Bit Protocol (ABP).
 * The ABP is used to transfer data over a possibly 'lossy' communication channel.
 * Each datum send by the Sender has to be positively acknowledged by the Receiver.
 * In case of a negativer acknowledgement the datum is re-transmitted.
 *
 * In the following specification the data types 'Bit' and 'Data'
 * are considered to be explicitly defined.
 */

/*
 * Defines the sender of the Alternating Bit Protocol
 */
process Sender {

    /* definition of actions, public entities are visible outside the Sender context */
    public read(Data)       /* action that reads a datum form the external channel */
    public send(Bit, Data)  /* transfers information to the internal forward comm. channel */
    public read_ack(Bit)    /* receives information from the internal backward comm. channel */
    public read_err()       /* receives an error from the internal backward comm. channel */


    /* definition of the publicly visible process cf. constructor */
    public Sender {
        Sender(0);      /* first send a 0 along with the data */
        Sender(1);      /* then send a 1 along with the data */
        Sender          /* ... and start all over again */
    }

    /* definition of auxiliary helper processes, b is the bit to be bundled with the datum */
    private Sender(b: Bit) {
        read(d: Data);      /* read a datum from the external channel */
        Sender(b, d);       /* transfer control to a further auxiliary process */
    }

    /* b is the bit to be bundled with the datum, d is the datum itself */
    private Sender(b: Bit, d: Data) {
        send(b, d);         /* send the datum along with the 'check' bit */
        WaitForAck(b, d)    /* ... and wait for the acknowledgement */
    }

    private WaitForAck(b: Bit, d: Data) {
        ALT {
        [] read_ack(b); OK                  /* receive a positive acknowledgement */
        [] read_ack(1 - b); Sender(b, d)    /* receive a negative acknowledgement and re-send */
        [] read_err(); Sender(b,d)          /* receive an error and re-send */
        }
    }
}
```

Literatur

[BBR10] Baeten, J. C. M., Basten, T., Reniers, M. A.: Process algebra (equational theories of communicating processes), vol. 50 of Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, Cambridge, 2010.

[BHk89] Bergstra, J.A., Heering, J., Klint, P., The algebraic specification formalism ASF, in: Algebraic specification, J.A. Bergstra, J. Heering & P. Klint (eds.), pp. 1-66, ACM Press Frontier Series, Addison-Wesley 1989.

[BK85] Bergstra, J. A., Klop, J. W.: Algebra of communicating processes with abstraction, Theoretical Computer Science, 37(1), 1985, 77–121.

[BV14] Breer, A., Veltink, G. J.: Modellierung von parallelen und verteilten Systemen mit domänenspezifischen Sprachen, Proceedings EKA 2014, Magdeburg, 2014.

[Hoa78] Hoare, C. A. R.: Communicating sequential processes, Communications of the ACM, 21(8), 1978, 666–677.

[Mil80] Milner, R.: A Calculus of Communicating Systems, vol. 92 of LNCS, 1980.

[MV90] Mauw, S., Veltink, G. J.: A process specification formalism, Fundam. Inf., 13(2), 1990, 85–139, ISSN 0169-2968.

[Vel10] Veltink, G. J.: PSF - A Retrospective, Fundam. Inf., 100 (1), 2010, 181–227, ISSN 0169-2968.