D-CON 2010 Themenvorschläge (Stand 26.1.2010, 17:00 Uhr)

1 Nebenläufigkeit und Kausalität von Interaktionen offener Systeme
(Tobias Heindel, Universität Duisburg Essen)

Geschlossene Systeme werden seit langem erfolgreich mit (Generalisierungen von) Petrinetzen modelliert. Nebenläufigkeit und Kausalität von Ereignissen in diesen Modellen sind gut verstanden und werden bei der Verifikation mit Partialordnungstechniken ausgenutzt. Für offene Systeme gibt es aber mehr offene Fragen als etablierte Theorie. Ein Erklärungsversuch dafür ist, dass die aktuellen Methoden aus der Welt der Prozesskalküle die Interaktionen eines offenen mit seiner Umwelt grundsätzlich verschieden betrachten: es ist im wesentlichen egal welcher Typ von Ereignis jeder Interaktion zu Grunde liegt, es kommt darauf an, welche Ressourcen und Informationen ausgetauscht werden.
Zu Beginn dieser Themenrunde werde ich kurz am Beispiel von Petrinetzen (als spezielle Graphtransformationssysteme) erläutern, welche Probleme auftreten, wenn man versucht die klassische Nebenläufigkeitstheorie für offene Systems zu entwickeln. Die zwei Hauptprobleme sind, dass man frei von Annahmen über die Umgebung sein sollte, und dass Kausalität und Nebenläufigkeit Informationen über das innere eines black-box Systems preisgeben könnte. Ich werde weiter kurz skizzieren, wie die gleichen Problematiken auch allgemein für reaktive Systeme im Sinne von Leifer und Milner auftreten -- aber nur so weit bis genug Grundlage geschaffen ist, für eine Diskussion alternativer Konzepte und verwandter Probleme.

---

2 Synchrone und asynchrone Interaktion in verteilten Systemen
Jens-Wolfhard Schicke

Für die Modellierung verteilter Systeme existieren inzwischen verschiedenste Ansätze.
Viele dieser Ansätze gehen von synchroner Kommunikation zwischen Systemkomponenten aus, andere Ansätze nehmen an, dass Nachrichten nur asynchron verschickt werden können.
Aus dem Bereich der verteilten Algorithmen ist bereits bekannt, dass bestimmte Probleme nur durch synchrone Interaktion der beteiligten Komponenten gelöst werden können.
Es scheint also bestimmte Muster in den Kommunikationen zu geben, die asynchron nicht abgebildet werden können.
Wir sind auf der Jagd nach solchen Mustern. Damit uns keine entwischen, suchen wir innerhalb verschiedener Beschreibungsmöglichkeiten eines Systems nach ihnen, konkret in Petrinetzen und im Pi-Kalkül Jungle. Um die Muster korrekt erkennen zu können haben wir uns von einfachen Interleaving-Äquivalenzen inzwischen großteils verabschiedet und verwenden jetzt "true-concurrency"-Semantiken teilweise auch unter Berücksichtigung von Kausalität.
Unser großer Traum ist eine exakte Grenzlinie zwischen Synchronität und Asynchronität irgendwo im linear-time branching-time Spektrums quer durch die verschiedenen Beschreibungsmöglichkeiten für Systeme. Für Petri Netze und branching-time haben wir inzwischen eine Grenze fast exakt beschrieben.

---

3 Parallelisation of Decision Procedures

Alexander Ditter, Jan Tobias Muehlberg, Uni Bamberg

In recent years a major shift in hardware architectures towards multi-core and many-core systems has taken place. In this session we want to discuss recent developments and future options for exploiting these architectures to efficiently solve decision problems. Starting off with a talk outlining the state of the art in parallelised SAT-solving and SMT-solving. Topics of the subsequent discussion may range from personal experience reports to elaborate exploration of technical challenges. We are especially interested in the exploration of shared memory CPU and GPU multi-processor systems. We are open for suggestions and contributions from D-CON participants.

---

4 Concurrency Theory in the Analysis of Wireless Sensor Systems

Christian Eisentraut (Universität des Saarlandes)

In this session concurrency theory meets concurrency practice. A consortium of companies and universities in the Netherlands has developed a wireless sensor system named MyriaNed. The sensor nodes in such a systems come equipped with a time synchronization algorithm and communicate in a time-slotted fashion (similar to the Zigbee protocol) by broadcasting messages. The question is now: can we use concurrency theory to model these wireless sensor nodes. More importantly, are the theoretical results reflected by the practical implementation? We will put this to the test by applying concurrency theory to concurrent programs running on a physical wireless sensor network.

Ansprechpartner: Holger Hermanns

---

5 Verifizierte Analysen für Nebenläufige Programme

Arbeitsgruppe: Müller-Olm, Münster

Session-Leiter: Peter Lammich

In letzter Zeit haben wir einige Analysen für nebenläufige Programme
mit Hilfe des Theorembeweisers Isabelle/HOL verifiziert, unter anderem Vorgänger-Berechnungen (pre^*) für DPNs und acquisition-history basierte Analysen für Programme mit Locks. In dieser Session sollen Möglichkeiten diskutiert werden, wie man aus den Formalisierungen für diese (und andere) Analysen effizienten, ausführbaren und korrekten Code ableiten kann. Eine Möglichkeit besteht darin, die Analysen als funktionale Programme im funktional Teil von Isabelle/HOL zu formalisieren, und daraus Haskell oder ML Code zu extrahieren. Tools wie das Isabelle Collection Framework stellen dafür notwendige, effiziente funktionale Datenstrukturen zur Verfügung. Eine weitere interessante Möglichkeit ist, die Analysen als Systeme von speziellen Horn-Klauseln zu formulieren (z.Bsp. Datalog). Solche Horn-Klausel basierten Formulierungen lassen sich in Isabelle/HOL elegant einbetten, und mit Hilfe von symbolischen Interpretern (z.Bsp. bddbddb) auch effizient ausführen.

---

6 Hierarchische Dekomposition von hybriden Systemen
Sessionleiter: Sven Schneider
Universität: TU-Berlin
Gruppe: Uwe Nestmann

Hybride Systeme (also beispielsweise Produktionsanlagensteuerungen) bestehen aus einem diskreten Algorithmus und sind in ein System mit komplexem Zeitverhalten eingebettet.
Durch die Kombination beider Aspekte sind keine Techniken einer der beiden Domänen zur Synthese/Verifikation direkt anwendbar.
Wir untersuchen den Ansatz der vertikalen/horizontalen Dekomposition des Problems und die komponentenbasierte Konstruktion/Spezifikation/Verifikation des Systems. Monolithische Entwicklungen für solche Algorithmen können zwar optimaler sein aber sind in realistischen Szenarien nicht effektiv verifizierbar. Aktuelles Problem ist die Definition der Komposition von Komponenten und deren Spezifikationen, so dass die Erfüllung der Spezifikation unter Komposition bewahrt wird. Für die formale Analyse definieren wir die genannten Begriffe möglichst sinnvoll für einen hybriden Prozesskalkül. Bei der angestrebten Methodik wird der einschränkende Einfluss der diskreten Reglerkomponenten auf das zu regelnde System mit der Beschreibung des Systemverhaltens schrittweise verschmolzen. Die dabei entstehenden Einschränkungen garantieren dann (hoffentlich) die Einhaltung der Spezifikation. Neben so verifizierbaren Sicherheitsaspekten muss noch beispielsweise eine Form von Nichtterminierung (Nonzenoness) überprüft werden.

Anmerkung: Sollte sich für dieses Thema jemand interessieren, der mit hybriden Systemen und eventuell sogar Prozesskalkülen vertraut ist, wäre ich auch sehr an einer 2-Personen-Session interessiert.

---

7 Verifikation von Safety und Livenesseigenschaften verteilter Algorithmen
Philipp Küfner (TU Berlin)

Konkret geht es hier um den <>S Consensus-Algorithmus von Chandra/Toueg, für den wir Validity, Agreement und Termination formal (mit Theorembeweiser Isabelle/HOL) bewiesen haben. Es soll dabei aber weniger um die Details des Isabelle/HOL Beweises gehen, sondern vielmehr um den jeweils für die einzelnen Eigenschaften gewählten Beweisstil, der sich für die Safety Eigenschaften (Validity und Agreement) wesentlich unterscheidet von dem Stil des Termination Beweises. Während es für die Safety-Eigenschaften reicht die Eigenschaften als Invarianten des Algorithmus nachzuweisen, ist dies für die Livenesseigenschaft nicht möglich. Wir haben schließlich unser Modell um explizit um runs Beweise über Runs geführt. Die Wesentliche (natürlich nicht vollständig beantwortete) Frage ist dabei natürlich, gibt es geschicktere Möglichkeiten im Hinblick auf die Wahl des zugrunde liegenden Modells und der Beweismethode.

---

8 From parallelism of expressions to concurrent executions of dataflow functional programs (work in progress).
Joaquin Aguado, Uni Bamberg

This practical talk presents some of the available parallel programming mechanisms of Haskell. In particular the focus is on (i) the semi-explicit parallelism codified by annotations and (ii) the (nested) data parallelism. The main motivation is to open the discussion on which of these parallelism abstractions (if any) fits better for a Dataflow (stream-based) lazy lambda-calculus model of computation.

---

9 Operational Partial Order Semantics for Relaxed Memory Consistency Models
Christian Eisentraut (Universität des Saarlandes, Prof. Dr. Holger Herrmanns)
Malte Lochau (TU Braunschweig, Prof. Dr. Ursula Goltz)

Multi threaded programming with shared memory access is a straightforward generalization of sequential programming, and thus a wide-spread and heavily used paradigm for concurrent programming in an industrial context. Programming languages implementing this approach include, beside others, Java and C#. The part of the semantics of a programming language exclusively concerned with the interaction of threads with memory is called the memory model. In the context of hardware and software optimizations that are usually only sound for purely sequential programs, these semantics are not clear for real world languages. More often than not, it deviates extensively from pure interleaving semantics. But this makes it hard for the average programmer to write correct programs. As a consequence, today's memory models need to prove of value in the contradictory contexts of both optimizations and programmer friendly semantics. Different relaxations from pure interleaving semantics can be described as partial orders between the instructions inside the single threads and a partial order between synchronization instructions between threads. However, most models fail to do this in an operational fashion, which we estimate a necessary condition for models that can be understood by the average programmer and allow for (semi-)automated analysis techniques. We hope that ideas from classical partial order semantic models, as for example event structures and Petri nets, which combine partial orders and operational semantics, can help to develop operational partial order semantics for memory models, and also to formalize concepts like causality of actions and out-of-thin-air reads.

In this session, we will briefly present existing approaches and then discuss new ideas, how operational partial order semantics can be realized for relaxed memory models.
