Handout 1 General info & topics

Issued: 22. 01. 2015

The lecturers of this course are

<table>
<thead>
<tr>
<th>lecturer</th>
<th>office</th>
<th>phone</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volker Stolz</td>
<td>(8462)</td>
<td>55 58 75 86 <a href="mailto:stolz@ifi.uio.no">stolz@ifi.uio.no</a> / <a href="mailto:volker.stolz@hib.no">volker.stolz@hib.no</a></td>
<td></td>
</tr>
<tr>
<td>Martin Steffen</td>
<td>8162</td>
<td>22 85 0417 <a href="mailto:msteffen@ifi.uio.no">msteffen@ifi.uio.no</a></td>
<td></td>
</tr>
</tbody>
</table>

The webpage of the course is found under

http://www.uio.no/studier/emner/matnat/ifi/INF5140/v15/

It contains information about the course, slides will be available, exercises, announcements, info about obligs, etc. A proposed weekly schedule is already there, but it may undergo adaptations as we go. If slides for forthcoming lectures have not yet been uploaded, you may find corresponding slides under the corresponding link for earlier versions of this course, as there will be a large overlap (but also adaptations).

1 Possible presentation topics

Part of the course will be “lecture style”, i.e., we present material getting some basics covered. The other part of the course will be covered by participants in the form of specialized topics.

1.1 Bounded model checking

Bounded model checking is a particular approach to address (or ignore) the fact that often systems are infinite. The starting point is to restrict the exploration of the model only up a given bound of steps or distance for the starting point(s) of the exploration. This bound may be chosen arbitrarily (as far a memory or time constraints allow) and may be also iteratively increased, but it’s given nonetheless (unlike for ordinary model checking which is done without a priori given bound). This boundedness of the reachability problem (“a bad state is reachable within n steps”) allows to represent the algorithm as a form of SAT solving. [Clarke et al., 2001]
### 1.2 Cache coherence with TLA+

TLA is a specification language invented and promoted by L. Lamport. It has been used for model checking as well. The talk could present the logic itself, the corresponding model checking tool and its use to check cache coherence. [Joshi et al., 2003]. You may also visit the web-page [http://esearch.microsoft.com/en-us/um/people/lamport/tla/tla.html](http://esearch.microsoft.com/en-us/um/people/lamport/tla/tla.html)

### 1.3 CTL model checking/ symbolic model checking

Branching time logic is a (family of) temporal logics, which are an alternative to the logic we mostly covered here, namely linear time temporal logic. CTL and its variants are prominent representatives of that family, and also used in model checkers such as SMV. The change of logics also entails that the underlying algorithm for checking is different. The talk should present the logics itself, compare CTL with LTL of the course, perhaps cover CTL* as one well-known variant, and show how model checking of CTL works. Often, CTL uses BDDs as an efficient representation, but that may be covered in a separate presentation. See [Clarke et al., 1999, Chap \subseteq 3, 4.1].

### 1.4 BDD model checking

Binary decision diagrams are one particularly efficient way of representing certain kind of data, which found applications in especially CTL model checking (for instance SMV). See [Clarke et al., 1999, Chapter 5]. The topic might be combined with CTL model checking.

### 1.5 SMV model checking of safety critical system

SVM is one very well known model checker, which is based on different principles than Spin, which we mostly use in the lecture part of the course. The principles may be covered in a separate, earlier presentation (see Sec. in 1.3). The talk should cover the tool and the case study [Kostimie, 2008]. It would be good if CTL/BDD etc are then covered by someone else.

### 1.6 Model checking of security protocols with CSP and FDR

CSP is one very prominent example of so-called process algebras, which can be seen as a specification formalism for parallel systems, and also used in context with model checking. FDR is one known tool for that. The topic may be split into 2 separate presentation, one presenting CSP and perhaps its semantics and other aspects. The second one concentrates on FDR and how to model check in particular security protocols with it. See [Ryan and Schneider, 2001].

### 1.7 Model checking of the modal $\mu$-calculus

A very expressive modal/temporal logic (typically from the branching time family) is the (modal/propositional) $\mu$-calculus, which is know for its expressivity and the explicit reliance of fix-points. The talk would present that logic, perhaps highlight connections to the other formalisms and present algorithmic aspects of model checking for it (for instance, Tableau-based model checking). [Cleaveland, 1990, Emerson, 1997]
1.8 Run-time verification

Run-time verification is an approach where properties are specified e.g. in LTL or as regular expressions, and then checked at runtime. This means that this is not verification in the strict sense of the word, as only a single run is explored each time. It thus more corresponds to a form of testing. Nonetheless, harnessing the power of temporal specification mechanisms has become a hot topic, and is the focus of the EU COST Action IC1402 “Monitoring beyond Runtime Verification” that also UiO and HiB take part in. In this talk, you will give an overview about the different notations, algorithms, and tools that are used in this area. [Leucker and Schallhart, 2009] [D’Angelo et al., 2005]

1.9 Group ware verification

Group ware refers to software and underlying protocols for distributed systems to allow computer supported cooperative work (CSCW). The talk could present some case study where model checking has been used to verify core ingredients for such protocols, for instance in [ter Beek et al., 2005] [ter Beek et al., 2009].

References


