Work Assignments for "Modal Logics" lectures

L. Caires

Abstract

I suggest two alternative topics. One is related to an application of modal logic to object-oriented program analysis, while the other investigates the decidability of validity in a logic for a fragment of CCS. The answer to the assignments are expected to be in the form a short research paper (no more than 15 pages). Please feel free to contact me to discuss the proposals or candidate solutions (whatever preliminary they might be).

1 Assignment A

This project explores the use of a modal logic ML to verify usage policies for objects in programs written in a toy object oriented language OL. For example, one may want to ensure that a certain method is called exactly once on a given object, and before any other method call on the same object: this might be useful if the method is an initialization operation. One may also want to impose that a file object may only be read after being open. The abstract syntax of OL, which is an expression language, is defined as follows

$$\begin{array}{rcl} E & ::= & \texttt{class}\ \{\mathbf{m}_1(x_1) = E_1, \ldots, \mathbf{m}_k(x_k) = E_k\} \\ & | & \texttt{let}\ x = E\ \texttt{in}\ E \\ & | & x.\mathbf{m}(y) \\ & | & \texttt{new}\ y \end{array}$$

Values of expressions are either class references (obtained by evaluating class expressions) or object references (obtained by evaluating new expressions). The class and object references may be modeled by pure names at the level of the language semantics. Notice that classes are first-class entities in OL, we can think of classes as (parameter-less) object generating functions. Evaluation of expressions is performed in the expected way, e.g., in a let expression $\texttt{let}\ x = E_1\ \texttt{in}\ E_2$ the expression E_1 is evaluated before E_2 (call by value). We do not formalize here the semantics of language, such a task is part of your work, although we expect computations of OL programs to be represented by sequences of program states with a certain structure. Here is an example of a program in OL:

$$\begin{array}{rl} P \triangleq & \texttt{let}\ c = \texttt{class}\ \{\mathbf{l}(x) = x, \mathbf{k}(x) = x\}\ \texttt{in} \\ & \quad \texttt{let}\ y = \texttt{new}\ c\ \texttt{in} \\ & \quad\quad \texttt{let}\ z = y.\mathbf{l}(y)\ \texttt{in} \\ & \quad\quad\quad \texttt{let}\ u = z.\mathbf{k}(y)\ \texttt{in}\ u \end{array}$$

The value of this program is a reference to the (unique) object it creates while running. We also consider the modal logic ML, with the following syntax:

$$A, B \quad ::= \quad \top \mid A \wedge B \mid \neg A \mid (x.\mathbf{m})A \mid \texttt{OB}\ x.A \mid \Diamond A$$

The logic ML is to be interpreted on sequences of program states, as follows. The propositional operators are understood in the standard way. The formula $\langle x.\mathbf{m}\rangle\, A$ says that the next method

to be invoked on object x is $\mathbf{m}$ and after that formula A holds. The quantifier $\texttt{OB}\ x.A$ existentially quantifies over the object references refering to the objects that are currently allocated. The formula $\diamond A$ asserts that the property A will eventually hold.

We can verify that the program P above satisfies the formula $\diamond\texttt{OB}x.(x.\mathbf{l})(x.\mathbf{k})\top$, and also the formula $\diamond\texttt{OB}x.(x.\mathbf{k})\top$. We can also verify that the program P does not satisfy the formula $\diamond\texttt{OB}x.(x.\mathbf{k})(x.\mathbf{l})\top$. In this project, you are asked to work out in detail the operational semantics of the object language OL and the semantics of the modal logic ML, in two different ways.

1. Define a notion of model appropriate to represent computation sequences (our programs are deterministic) in our object-oriented language OL, namely, a suitable class of labeled transition systems. Define the operational semantics of the object-oriented language by showing how to assign to each program a suitable LTS. An appropriate way to do that will be by means of a set of SOS rules (see Rocco De Nicola's lectures).

2. Precisely define the semantics of our modal logic ML, by interpreting its formulas on the LTSs you have just defined in 1, by means of a satisfaction relation $\models$ defined between states and formulas.

3. Define an alternative (but equivalent) semantics for our object oriented language by means of an encoding in the π-calculus. Hint: you may define a mapping $C[\![-]\!]_x$ translating every expression E of the object-oriented language into a π-calculus process $C[\![E]\!]_x$ that outputs the result of evaluating E on a given channel name x (a name fresh in E).

4. Define an embedding $L[\![-]\!]$ that translates any formula A of our modal logic into a formula $L[\![A]\!]$ of the pi-calculus spatial logic discussed in the lectures, such that the following property holds:

$$E \models A \textit{ if and only if } C[\![E]\!]_x \models L[\![A]\!]$$

5. Prove your results and illustrate your encodings in a couple of examples using the Spatial Logic Model Checker.

2 Assignment B

Consider the modal process logic

$$A ::= \neg A \mid A \wedge B \mid \mathbf{0} \mid A \mid B \mid A \triangleright B \mid \langle a \rangle A$$

where the action a mentioned in $\langle a \rangle A$ is not the internal action τ. The logic is to be interpreted as expected in the simple CCS model adopted in Caires and Lozes "Elimination of Quantifiers and Undecidability in Spatial Logics for Concurrency" (Concur 2004).

The main goal of this project is to determine decidability of model checking for this spatial logic, even if it contains the guarantee operator $A \triangleright B$. You may start to prove that this particular combination of logical operators and semantics satisfy a "small model" property (Hint: find a notion of bisimilarity $\sim$ such that $P \sim_k Q$ and $P \models A$ implies $Q \models A$ for any formula A of "size" less than k. Show that for any process P, we may find a process Q of "size" less that k such that $P \sim_k Q$. A brute force algorithm can then model-check $A \mid B$ by model enumeration, although you may want to think about something better).