* (a) $\mathrm{AG}\,(\phi \wedge \psi) \equiv \mathrm{AG}\,\phi\ ?\ \mathrm{AG}\,\psi$
(b) $\mathrm{EF}\,\neg\phi \equiv \neg??\phi$

12. State explicitly the meaning of the temporal connectives AR etc., as defined on page 217.
13. Prove the equivalences (3.6) on page 216.
* 14. Write pseudo-code for a recursive function TRANSLATE which takes as input an arbitrary CTL formula ϕ and returns as output an equivalent CTL formula ψ whose only operators are among the set $\{\bot, \neg, \wedge, \mathrm{AF}, \mathrm{EU}, \mathrm{EX}\}$.

Exercises 3.5

1. Express the following properties in CTL and LTL whenever possible. If neither is possible, try to express the property in CTL*:
* (a) Whenever p is followed by q (after finitely many steps), then the system enters an 'interval' in which no r occurs until t.
(b) Event p precedes s and t on all computation paths. (You may find it easier to code the negation of that specification first.)
(c) After p, q is never true. (Where this constraint is meant to apply on all computation paths.)
(d) Between the events q and r, event p is never true.
(e) Transitions to states satisfying p occur at most twice.
* (f) Property p is true for every second state along a path.
2. Explain in detail why the LTL and CTL formulas for the practical specification patterns of pages 183 and 215 capture the stated 'informal' properties expressed in plain English.
3. Consider the set of LTL/CTL formulas $\mathcal{F} = \{\mathrm{F}\,p \rightarrow \mathrm{F}\,q, \mathrm{AF}\,p \rightarrow \mathrm{AF}\,q, \mathrm{AG}\,(p \rightarrow \mathrm{AF}\,q)\}$.
(a) Is there a model such that all formulas hold in it?
(b) For each $\phi \in \mathcal{F}$, is there a model such that ϕ is the only formula in $\mathcal{F}$ satisfied in that model?
(c) Find a model in which no formula of $\mathcal{F}$ holds.
4. Consider the CTL formula $\mathrm{AG}\,(p \rightarrow \mathrm{AF}\,(s \wedge \mathrm{AX}\,(\mathrm{AF}\,t)))$. Explain what exactly it expresses in terms of the order of occurrence of events p, s and t.
5. Extend the algorithm `NNF` from page 62 which computes the negation normal form of propositional logic formulas to CTL*. Since CTL* is defined in terms of two syntactic categories (state formulas and path formulas), this requires two separate versions of `NNF` which call each other in a way that is reflected by the syntax of CTL* given on page 218.
6. Find a transition system which distinguishes the following pairs of CTL* formulas, i.e., show that they are not equivalent:
(a) $\mathrm{AF}\,\mathrm{G}\,p$ and $\mathrm{AF}\,\mathrm{AG}\,p$
* (b) $\mathrm{AG}\,\mathrm{F}\,p$ and $\mathrm{AG}\,\mathrm{EF}\,p$
(c) $\mathrm{A}[(p\ \mathrm{U}\ r) \vee (q\ \mathrm{U}\ r)]$ and $\mathrm{A}[(p \vee q)\ \mathrm{U}\ r)]$

* (d) $\mathrm{A}[X\,p \vee \mathrm{X}\,\mathrm{X}\,p]$ and $\mathrm{AX}\,p \vee \mathrm{AX}\,\mathrm{AX}\,p$
(e) $\mathrm{E}[\mathrm{G}\,\mathrm{F}\,p]$ and $\mathrm{EG}\,\mathrm{EF}\,p$.

7. The translation from CTL with boolean combinations of path formulas to plain CTL introduced in Section 3.5.1 is not complete. Invent CTL equivalents for:
* (a) $\mathrm{E}[\mathrm{F}\,p \wedge (q\ \mathrm{U}\ r)]$
* (b) $\mathrm{E}[\mathrm{F}\,p \wedge \mathrm{G}\,q]$.
In this way, we have dealt with all formulas of the form $\mathrm{E}[\phi \wedge \psi]$. Formulas of the form $\mathrm{E}[\phi \vee \psi]$ can be rewritten as $\mathrm{E}[\phi] \vee \mathrm{E}[\psi]$ and $\mathrm{A}[\phi]$ can be written $\neg\mathrm{E}[\neg\phi]$. Use this translation to write the following in CTL:
(c) $\mathrm{E}[(p\ \mathrm{U}\ q) \wedge \mathrm{F}\,p]$
* (d) $\mathrm{A}[(p\ \mathrm{U}\ q) \wedge \mathrm{G}\,p]$
* (e) $\mathrm{A}[\mathrm{F}\,p \rightarrow \mathrm{F}\,q]$.

8. The aim of this exercise is to demonstrate the expansion given for AW at the end of the last section, i.e., $\mathrm{A}[p\ \mathrm{W}\ q] \equiv \neg\mathrm{E}[\neg q\ \mathrm{U}\ \neg(p \vee q)]$.
(a) Show that the following LTL formulas are valid (i.e., true in any state of any model):
(i) $\neg q\ \mathrm{U}\ (\neg p \wedge \neg q) \rightarrow \neg\mathrm{G}\,p$
(ii) $\mathrm{G}\,\neg q \wedge \mathrm{F}\,\neg p \rightarrow \neg q\ \mathrm{U}\ (\neg p \wedge \neg q)$.
(b) Expand $\neg((p\ \mathrm{U}\ q) \vee \mathrm{G}\,p)$ using de Morgan rules and the LTL equivalence $\neg(\phi\ \mathrm{U}\ \psi) \equiv (\neg\psi\ \mathrm{U}\ (\neg\phi \wedge \neg\psi)) \vee \neg\mathrm{F}\,\psi$.
(c) Using your expansion and the facts (i) and (ii) above, show $\neg((p\ \mathrm{U}\ q) \vee \mathrm{G}\,p) \equiv \neg q\ \mathrm{U}\ \neg(p \wedge q)$ and hence show that the desired expansion of AW above is correct.

Exercises 3.6

* 1. Verify ϕ_1 to ϕ_4 for the transition system given in Figure 3.11 on page 198. Which of them require the fairness constraints of the SMV program in Figure 3.10?
2. Try to write a CTL formula that enforces non-blocking and no-strict-sequencing at the same time, for the SMV program in Figure 3.10 (page 196).
* 3. Apply the labelling algorithm to check the formulas ϕ_1, ϕ_2, ϕ_3 and ϕ_4 of the mutual exclusion model in Figure 3.7 (page 188).
4. Apply the labelling algorithm to check the formulas ϕ_1, ϕ_2, ϕ_3 and ϕ_4 of the mutual exclusion model in Figure 3.8 (page 191).
5. Prove that (3.8) on page 228 holds in all models. Does your proof require that for every state s there is some state s' with $s \rightarrow s'$?
6. Inspecting the definition of the labelling algorithm, explain what happens if you perform it on the formula $p \wedge \neg p$ (in any state, in any model).
7. Modify the pseudo-code for `SAT` on page 227 by writing a special procedure for $\mathrm{AG}\,\psi_1$, without rewriting it in terms of other formulas[5].

[5] Question: will your routine be more like the routine for AF, or more like that for EG on page 224? Why?

* 8. Write the pseudo-code for $\texttt{SAT}_{\text{EG}}$, based on the description in terms of deleting labels given in Section 3.6.1.

* 9. For mutual exclusion, draw a transition system which forces the two processes to enter their critical section in strict sequence and show that ϕ_4 is false of its initial state.

10. Use the definition of $\models$ between states and CTL formulas to explain why $s \models \text{AG AF}\,\phi$ means that ϕ is true infinitely often along every path starting at s.

* 11. Show that a CTL formula ϕ is true on infinitely many states of a computation path $s_0 \to s_1 \to s_2 \to \ldots$ iff for all $n \geq 0$ there is some $m \geq n$ such that $s_m \models \phi$.

12. Run the NuSMV system on some examples. Try commenting out, or deleting, some of the fairness constraints, if applicable, and see the counter examples NuSMV generates. NuSMV is very easy to run.

13. In the one-bit channel, there are two fairness constraints. We could have written this as a single one, inserting '&' between `running` and the long formula, or we could have separated the long formula into two and made it into a total of three fairness constraints.
    In general, what is the difference between the single fairness constraint $\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n$ and the n fairness constraints $\phi_1, \phi_2, \ldots, \phi_n$? Write an SMV program with a fairness constraint `a & b` which is not equivalent to the two fairness constraints `a` and `b`. (You can actually do it in four lines of SMV.)

14. Explain the construction of formula ϕ_4, used to express that the processes need not enter their critical section in strict sequence. Does it rely on the fact that the safety property ϕ_1 holds?

* 15. Compute the $\text{E}_C\text{G}\,\top$ labels for Figure 3.11, given the fairness constraints of the code in Figure 3.10 on page 196.

Exercises 3.7

1. Consider the functions

$$H_1, H_2, H_3 \colon \mathcal{P}(\{1,2,3,4,5,6,7,8,9,10\}) \to \mathcal{P}(\{1,2,3,4,5,6,7,8,9,10\})$$

defined by

$$\begin{aligned} H_1(Y) &\stackrel{\text{def}}{=} Y - \{1,4,7\} \\ H_2(Y) &\stackrel{\text{def}}{=} \{2,5,9\} - Y \\ H_3(Y) &\stackrel{\text{def}}{=} \{1,2,3,4,5\} \cap (\{2,4,8\} \cup Y) \end{aligned}$$

for all $Y \subseteq \{1,2,3,4,5,6,7,8,9,10\}$.

* (a) Which of these functions are monotone; which ones aren't? Justify your answer in each case.

* (b) Compute the least and greatest fixed points of H_3 using the iterations H_3^i with $i = 1, 2, \ldots$ and Theorem 3.24.

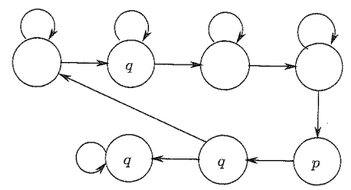

Figure 3.42. Another system for which we compute invariants.

(c) Does H_2 have any fixed points?

(d) Recall $G \colon \mathcal{P}(\{s_0, s_1\}) \to \mathcal{P}(\{s_0, s_1\})$ with

$$G(Y) \stackrel{\text{def}}{=} \texttt{if } Y = \{s_0\} \texttt{ then } \{s_1\} \texttt{ else } \{s_0\}.$$

Use mathematical induction to show that G^i equals G for all odd numbers $i \geq 1$. What does G^i look like for even numbers i?

* 2. Let A and B be two subsets of S and let $F \colon \mathcal{P}(S) \to \mathcal{P}(S)$ be a monotone function. Show that:
   (a) $F_1 \colon \mathcal{P}(S) \to \mathcal{P}(S)$ with $F_1(Y) \stackrel{\text{def}}{=} A \cap F(Y)$ is monotone;
   (b) $F_2 \colon \mathcal{P}(S) \to \mathcal{P}(S)$ with $F_2(Y) \stackrel{\text{def}}{=} A \cup (B \cap F(Y))$ is monotone.

3. Use Theorems 3.25 and 3.26 to compute the following sets (the underlying model is in Figure 3.42):
   (a) $[\![\text{EF}\,p]\!]$
   (b) $[\![\text{EG}\,q]\!]$.

4. Using the function $F(X) = [\![\phi]\!] \cup \text{pre}_\forall(X)$ prove that $[\![\text{AF}\,\phi]\!]$ is the least fixed point of F. Hence argue that the procedure $\texttt{SAT}_{\texttt{AF}}$ is correct and terminates.

* 5. One may also compute $\text{AG}\,\phi$ directly as a fixed point. Consider the function $H \colon \mathcal{P}(S) \to \mathcal{P}(S)$ with $H(X) = [\![\phi]\!] \cap \text{pre}_\forall(X)$. Show that H is monotone and that $[\![\text{AG}\,\phi]\!]$ is the greatest fixed point of H. Use that insight to write a procedure $\texttt{SAT}_{\texttt{AG}}$.

6. Similarly, one may compute $\text{A}[\phi_1\ \text{U}\ \phi_2]$ directly as a fixed point, using $K \colon \mathcal{P}(S) \to \mathcal{P}(S)$, where $K(X) = [\![\phi_2]\!] \cup ([\![\phi_1]\!] \cap \text{pre}_\forall(X))$. Show that K is monotone and that $[\![\text{A}[\phi_1\ \text{U}\ \phi_2]]\!]$ is the least fixed point of K. Use that insight to write a procedure $\texttt{SAT}_{\texttt{AU}}$. Can you use that routine to handle all calls of the form $\text{AF}\,\phi$ as well?

7. Prove that $[\![\text{A}[\phi_1\ \text{U}\ \phi_2]]\!] = [\![\phi_2 \vee (\phi_1 \wedge \text{AX}\,(\text{A}[\phi_1\ \text{U}\ \phi_2]))]\!]$.

8. Prove that $[\![\text{AG}\,\phi]\!] = [\![\phi \wedge \text{AX}\,(\text{AG}\,\phi)]\!]$.

9. Show that the repeat-statements in the code for $\texttt{SAT}_{\texttt{EU}}$ and $\texttt{SAT}_{\texttt{EG}}$ always terminate. Use this fact to reason informally that the main program `SAT` terminates for all valid CTL formulas ϕ. Note that some subclauses, like the one for AU, call `SAT` recursively and with a more complex formula. Why does this not affect termination?

3.9 Bibliographic notes

Temporal logic was invented by the philosopher A. Prior in the 1960s; his logic was similar to what we now call LTL. The first use of temporal logic for reasoning about concurrent programs was by A. Pnueli [Pnu81]. The logic CTL was invented by E. Clarke and E. A. Emerson (during the early 1980s); and CTL* was invented by E. A. Emerson and J. Halpern (in 1986) to unify CTL and LTL.

CTL model checking was invented by E. Clarke and E. A. Emerson [CE81] and by J. Quielle and J. Sifakis [QS81]. The technique we described for LTL model checking was invented by M. Vardi and P. Wolper [VW84]. Surveys of some of these ideas can be found in [CGL93] and [CGP99]. The theorem about adequate sets of CTL connectives is proved in [Mar01].

The original SMV system was written by K. McMillan [McM93] and is available with source code from Carnegie Mellon University[6]. NuSMV[7] is a reimplementation, developed in Trento by A. Cimatti, and M. Roveri and is aimed at being customisable and extensible. Extensive documentation about NuSMV can be found at that site. NuSMV supports essentially the same system description language as CMU SMV, but it has an improved user interface and a greater variety of algorithms. For example, whereas CMU SMV checks only CTL specification, NuSMV supports LTL and CTL. NuSMV implements bounded model checking [BCCZ99]. Cadence SMV[8] is an entirely new model checker focused on compositional systems and abstraction as ways of addressing the state explosion problem. It was also developed by K. McMillan and its description language resembles but much extends the original SMV.

A website which gathers frequently used specification patterns in various frameworks (such as CTL, LTL and regular expressions) is maintained by M. Dwyer, G. Avrunin, J. Corbett and L. Dillon[9].

Current research in model checking includes attempts to exploit abstractions, symmetries and compositionality [CGL94, Lon83, Dam96] in order to reduce the impact of the state explosion problem.

The model checker Spin, which is geared towards asynchronous systems and is based on the temporal logic LTL, can be found at the Spin website[10]. A model checker called FDR2 based on the process algebra CSP is available[11].

[6] www.cs.cmu.edu/~modelcheck/
[7] nusmv.irst.itc.it
[8] www-cad.eecs.berkeley.edu/~kenmcmil/
[9] patterns.projects.cis.ksu.edu/
[10] netlib.bell-labs.com/netlib/spin/whatispin.html
[11] www.fsel.com.fdr2_download.html

The Edinburgh Concurrency Workbench[12] and the Concurrency Workbench of North Carolina[13] are similar software tools for the design and analysis of concurrent systems. An example of a customisable and extensible modular model checking frameworks for the verification of concurrent software is Bogor[14].

There are many textbooks about verification of reactive systems; we mention [MP91, MP95, Ros97, Hol90]. The SMV code contained in this chapter can be downloaded from `www.cs.bham.ac.uk/research/lics/`.

[12] www.dcs.ed.ac.uk/home/cwb
[13] www.cs.sunysb.edu/~cwb
[14] http://bogor.projects.cis.ksu.edu/