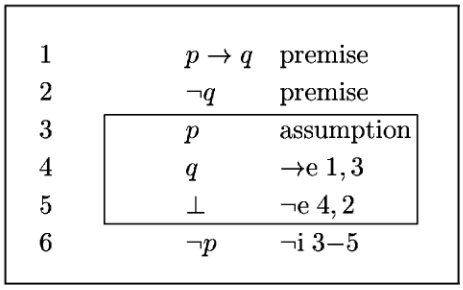
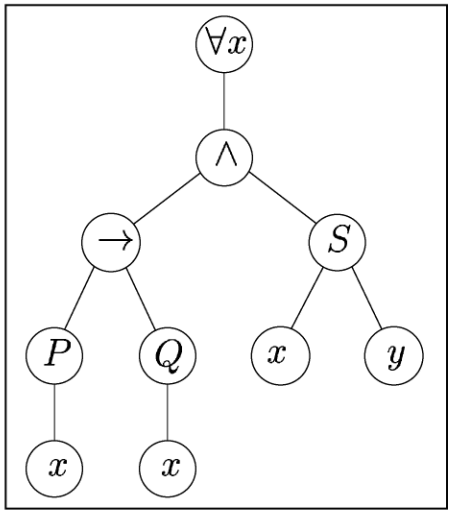
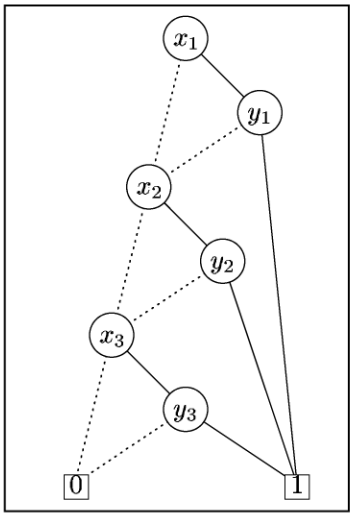
Second Edition
Logic in Computer Science:
Modelling and reasoning about systems
Solutions to designated exercises

Michael Huth and Mark Ryan



1	$p \to q$	premise
2	$\neg q$	premise
3	p	assumption
4	q	$\to$e 1, 3
5	$\bot$	$\neg$e 4, 2
6	$\neg p$	$\neg$i 3–5






```
k = 2;
t = a[1];
s = a[1];
while (k != n+1) {
   t = min(t+a[k], a[k]);
   s = min(s,t);
   k = k+1;
}
```

1(y). We prove the validity of $(p \wedge q) \vee (p \wedge r) \vdash p \wedge (q \vee r)$ by

1	$(p \wedge q) \vee (p \wedge r)$	premise
2	$p \wedge q$	assumption
3	p	$\wedge e_1$ 2
4	q	$\wedge e_2$ 2
5	$q \vee r$	$\vee i_1$ 4
6	$p \wedge (q \vee r)$	$\wedge$i 3, 5
7	$p \wedge r$	assumption
8	p	$\wedge e_1$ 7
9	r	$\wedge e_2$ 7
10	$q \vee r$	$\vee i_2$ 9
11	$p \wedge (q \vee r)$	$\wedge$i 8, 10
12	$p \wedge (q \vee r)$	$\vee$e $1, 2-6, 7-11$

2(a). We prove the validity of $\neg p \to \neg q \vdash q \to p$ by

1	$\neg p \to \neg q$	premise
2	q	assumption
3	$\neg\neg q$	$\neg\neg$i 2
4	$\neg\neg p$	MT 1, 3
5	p	$\neg\neg$e 4
6	$q \to p$	$\to$i $2-5$

$\neg\neg q$

$p \wedge q$

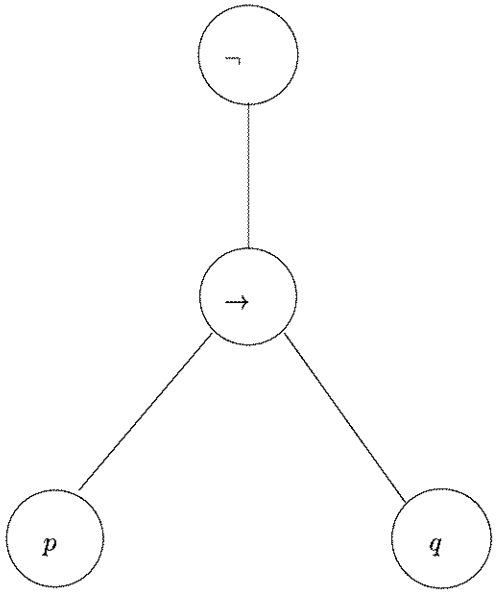
$\neg\neg q \to (p \wedge q)$

$\neg p \vee (\neg\neg q \to (p \wedge q))$

$p \to (\neg p \vee (\neg\neg q \to (p \wedge q)))$.

Note that $\neg q$ and $\neg\neg q$ are two different subformulas.

3(a). An example of a parse tree of a propositional logic formula which is a negation of an implication:



3(c). An example of a parse tree of a formula which is a conjunction of conjunctions is

$$\begin{aligned} &= [(n+1)+12]+(n+11) \\ &\geq (n+1)+12 \end{aligned}$$

guarantees that our claim holds for $n+1$. Note that the first step of this computation also uses that multiplication with a positive number is *monotone*: $x \geq y$ implies $2 \cdot x \geq 2 \cdot y$.

8. The Fibonacci numbers are defined by

$$\begin{aligned} F_1 &\stackrel{\text{def}}{=} 1 \\ F_2 &\stackrel{\text{def}}{=} 1 \\ F_{n+1} &\stackrel{\text{def}}{=} F_n + F_{n-1} \qquad \text{for all } n \geq 2. \end{aligned} \tag{1.5}$$

We use mathematical induction to prove that F_{3n} is even for all $n \geq 1$.

1. **Base Case:** For $n = 1$, we compute $F_{3\cdot 1} = F_3 = F_2 + F_1$ by (1.5), but the latter is just $1 + 1 = 2$ which is even.
2. **Inductive Step:** Our **induction hypothesis** assumes that F_{3n} be even. We need to show that $F_{3\cdot(n+1)}$ is even as well. This is a bit tricky as we have to decide on where to apply the inductive definition of (1.5):

$$\begin{aligned} F_{3\cdot(n+1)} &= F_{3n+3} \\ &= F_{(3n+2)+1} \\ &= F_{3n+2} + F_{(3n+2)-1} \qquad \text{by (1.5)} \\ &= F_{(3n+1)+1} + F_{3n+1} \\ &= (F_{3n+1} + F_{3n}) + F_{3n+1} \qquad \text{unfolding } F_{(3n+1)+1} \text{ with (1.5)} \\ &= 2 \cdot F_{3n+1} + F_{3n}. \end{aligned}$$

Since F_{3n} is assumed to be even and since $2 \cdot F_{3n+1}$ clearly is even, we conclude that $2 \cdot F_{3n+1} + F_{3n}$, and therefore $F_{3\cdot(n+1)}$, is even as well.
Note that it was crucial not to unfold F_{3n+1} as well; otherwise, we would obtain four summands but no inductive argument. (Why?)

10. Consider the assertion

"*The number* $n^2 + 5n + 1$ *is even for all* $n \geq 1$."

(a) We prove the **inductive step** of that assertion as follows: we

(v) $\times$; again, we can't nest predicates (B, S are predicates).
(vi) $\surd$
(vii) $\surd$
(viii) $\times$; again, we can't nest predicates.

4(a). The parsetree is

12. We prove the validity of $S \to \forall x\, Q(x) \vdash \forall x\, (S \to Q(x))$ by

1	$S \to \forall x\, Q(x)$	prem
2	x_0	
3	$\quad S$	assum
4	$\quad \forall x\, Q(x)$	$\to$e 1, 3
5	$\quad Q(x_0)$	$\forall x$ e 4
6	$S \to Q(x_0)$	$\to$i 3–5
7	$\forall x\, (S \to Q(x))$	$\forall x$ i 2–6

13(a). We show the validity of

$$\forall x\, P(a, x, x), \forall x\, \forall y\, \forall z\, (P(x, y, z) \to P(f(x), y, f(z))) \vdash P(f(a), a, f(a))$$

by

1	$\forall x\, P(a, x, x)$	prem
2	$\forall x\, \forall y\, \forall z\, (P(x, y, z) \to P(f(x), y, f(z)))$	prem
3	$P(a, a, a)$	$\forall x$ e 1
4	$\forall y\, \forall z\, (P(a, y, z) \to P(f(a), y, f(z)))$	$\forall x$ e 2
5	$\forall z\, (P(a, a, z) \to P(f(a), a, f(z)))$	$\forall y$ e 4
6	$P(a, a, a) \to P(f(a), a, f(a))$	$\forall z$ e 5
7	$P(f(a), a, f(a))$	$\to$e 6, 3

13(b). We show the validity of

$$\forall x P(a, x, x), \forall x\, \forall y\, \forall z\, (P(x, y, z) \to P(f(x), y, f(z))) \vdash \exists z\, P(f(a), z, f(f(a)))$$

by

1	$\forall x P(a, x, x)$	prem
2	$\forall x\, \forall y\, \forall z\, (P(x, y, z) \to P(f(x), y, f(z)))$	prem
3	$P(a, f(a), f(a))$	$\forall x$ e 1
4	$\forall y\, \forall z\, (P(a, y, z) \to P(f(a), y, f(z)))$	$\forall x$ e 2
5	$\forall z\, (P(a, f(a), z) \to P(f(a), f(a), f(z)))$	$\forall y$ e 4
6	$P(a, f(a), f(a)) \to P(f(a), f(a), f(f(a)))$	$\forall z$ e 5
7	$P(f(a), f(a), f(f(a)))$	$\to$e 6, 3
8	$\exists z\, P(f(a), z, f(f(a)))$	$\exists z$ i 7

```
fun Commutes (G: Group) {
  all a,b: G.elements | a.(b.(G.mult)) = b.(a.(G.mult))
} run Commutes for 3 but 1 Group

assert Commutative {
  all G: Group | Commutes(G)
} check Commutative for 5 but 1 Group
```

ii. Analyzing the assertion above we find no solution. The small-scope hypothesis therefore suggests that all finite groups are commutative. This time, the small-scope hypothesis got it wrong! There are finite groups that are not commutative.

iii. In fact, increasing the scope from 5 to 6 reveals a violation to our goal. Please run this analysis yourself and inspect the navigable tree to determine where and how commutativity is broken.

6(e) Yes, the assertions are formulas that make a claim about *all* groups. So a counter-example exists iff it exists for a single group. We already achieved the restriction to one group with the `but 1 Group` in the `check` and `run` directives.

end case
end function

You can now prove, by mathematical induction on the height of ϕ's parse tree, that the call TRANSLATE ϕ terminates and that the resulting formula has connectives only from the set $\{\bot, \neg, \wedge, \mathrm{AF}, \mathrm{EU}, \mathrm{EX}\}$.

EXERCISES 3.5 (p.250)

1(a). The process of translating informal requirements into formal specifications is subject to various pitfalls. One of them is simply ambiguity. For example, it is unclear whether "after some finite steps" means "at least one, but finitely many steps", or whether zero steps are allowed as well. It may also be debatable what "then" exactly means in "... then the system enters ...". We chose to solve this problem for the case when zero steps are not admissible, mostly since "followed by" suggests a real state transition to take place. The CTL formula we came up with is

$$\mathrm{AG}\,(p \rightarrow \mathrm{AX}\,\mathrm{AG}\,(\neg q \vee A[\neg r\ \mathrm{U}\ t]))$$

which in LTL may be expressed as

$$\mathrm{G}(p \rightarrow \mathrm{XG}(\neg q \vee \neg r\ \mathrm{U}\ t)).$$

It says: At any state, if p is true, then at any state which one can reach with at least one state transition from here, either q is false, or r is false until t becomes true (for all continuations of the computation path). This is evidently the property we intent to model. Variaous other "equivalent" solutions can be given.

1(f). The informal specification is ambiguous. Assuming that we mean a particular path we have to say that there exists some path on which p is true every second state (this is the global part). We assume that the informal description meant to set this off such that p is true at the first (=current), third etc. state. The CTL* formula thus reads as

$$\mathrm{E}[\mathrm{G}(p \wedge \mathrm{XX}\,p)].$$

Note that this is indeed a CTL* formula and you can check that it insists on a path $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \ldots$ where $s_0, s_2, s_4, \ldots$ all satisfy p.

- The function H_2 is not montone: e.g. $\{2\} \subseteq \{2,5\}$, but $H_2(\{2\})$, which is $\{5,9\}$, is not a subset of $H_2(\{2,5\})$, which is $\{9\}$.
- The function H_3 is monotone since union and intersection are monotone and the composition of monotone operations is again monotone.

1(b). We compute the least and greatest fixed-point of H_3 with just one iteration each: it is $\{2,4\}$ in each of these cases. So why can we say that this is the only fixed-point of H_3?

2(a) Let $X \subseteq X'$ and $s \in F_1(X)$. Then $s \in A \cap F(X)$ implies $s \in A$ and $s \in F(X)$. So in order to show $s \in F_1(X') = A \cap F(X')$ it suffices to show $s \in F(X')$. But this follows from $s \in F(X)$ as $X \subseteq X'$ and since F is monotone.

2(b) Let $X \subseteq X'$ and consider $s \in F_2(X)$.

- If $s \in A$, then $s \in A \cup (B \cap F(X') = F_2(X')$ follows.
- If $s \notin A$, then $s \in F_2(X) = A \cup (B \cap F(X))$ implies $s \in B \cap F(X)$, but we already saw in item (a) that this implies $s \in B \cap F(X')$ and so $s \in A \cup (B \cap F(X')) = F_2(X')$ does it.

5. • Let $X \subseteq X'$ and $s \in H(X) = [\![\phi]\!] \cap \{s_0 \in S \mid s_0 \to s_1 \text{ implies } s_1 \in X\}$. Then $s \in [\![\phi]\!]$ is clear. If $s' \in S$ with $s \to s'$, then $s \in H(X)$ implies $s' \in X$. Since $X \subseteq X'$, we infer from that $s' \in X'$. But this shows $s \in [\![\phi]\!] \cap \{s_0 \in S \mid s_0 \to s_1 \text{ implies } s_1 \in X'\} = H(X')$.

- We compute

$$\begin{aligned} H([\![\mathrm{AG}\,\phi]\!]) &= [\![\phi]\!] \cap \{s_0 \in S \mid s_0 \to s_1 \text{ implies } s_1 \in [\![\mathrm{AG}\,\phi]\!]\} \\ &= [\![\phi]\!] \cap [\![\mathrm{AX}\,\mathrm{AG}\,\phi]\!] \\ &= [\![\mathrm{AG}\,\phi]\!]. \end{aligned}$$

- We have already seen that $[\![\mathrm{AG}\,\phi]\!]$ is a fixed point of H. To show that it is the greatest fixed point, it suffices to show here that any set X with $H(X) = X$ has to be contained in $[\![\mathrm{AG}\,\phi]\!]$. So let s_0 be an element of such a fixed point X. We need to show that s_0 is in $[\![\mathrm{AG}\,\phi]\!]$ as well. For that we use the fact that

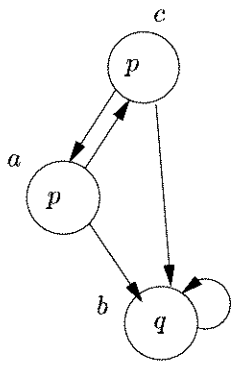
$$s_0 \in X = H(X) = [\![\phi]\!] \cap \{s \in S \mid s \to s_1 \text{ implies } s_1 \in X\}$$

to infer that $s_1 \in X$ for all s_1 with $s_0 \to s_1$. But, since s_1 is in X, we may apply that same argument to $s_1 \in X = H(X) = [\![\phi]\!] \cap \{s \in S \mid s \to s_2 \text{ implies } s_2 \in X\}$ and we get $s_2 \in X$ for all s_2 with $s_1 \to s_2$. By mathematical induction, we can therefore show that $s_n \in X$, where s_n is any state that is reachable from s_0. Since all states in X satisfy ϕ, this means that s is in $[\![\mathrm{AG}\,\phi]\!]$.

$\{\!\{x \geq 0\}\!\}$	
$\{\!\{(1 \cdot (x!) = x!) \wedge (x \geq 0)\}\!\}$	Implied
`a = x;`	
$\{\!\{(1 \cdot (a!) = x!) \wedge (a \geq 0)\}\!\}$	Assignment
`y = 1;`	
$\{\!\{(y \cdot (a!) = x!) \wedge (a \geq 0)\}\!\}$	Assignment
`while a > 0 {`	
$\{\!\{(y \cdot (a!) = x!) \wedge (a \geq 0) \wedge (a > 0)\}\!\}$	Invariant Hyp. $\wedge$ guard
$\{\!\{(y \cdot ((a-1)! \cdot a) = x!) \wedge (a - 1 \geq 0)\}\!\}$	Implied
$\{\!\{((y \cdot a) \cdot ((a-1)!) = x!) \wedge (a - 1 \geq 0)\}\!\}$	Implied
`y = y * a;`	
$\{\!\{(y \cdot ((a-1)!) = x!) \wedge (a - 1 \geq 0)\}\!\}$	Assignment
`a = a - 1;`	
$\{\!\{(y \cdot (a!) = x!) \wedge (a \geq 0)\}\!\}$	Assignment
`}`	
$\{\!\{(y \cdot (a!) = x!) \wedge (a \geq 0) \wedge \neg(a > 0)\}\!\}$	Partial-while
$\{\!\{y = x!\}\!\}$	Implied

Note that we had to strenghten our invariant hypothesis by adding a conjunct $(a \geq 0)$. This was needed since $\neg(a > 0)$ does not in itself imply the desired $(a = 0)$ to secure that our invariant implies the postcondition.

21(a). We simulate the code for each of these arrays in a table which lists



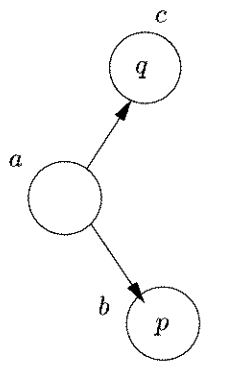
The world b satisfies $p \rightarrow \Box\Diamond q$, since $c \not\Vdash p$. For the worlds a and c, we have $a \Vdash p$ and $c \Vdash p$, so we need to secure $a \Vdash \Box\Diamond q$ and $c \Vdash \Box\Diamond q$. But this can be done, since all three worlds, x, of this model have an immediate successor state, x', such that $x' \Vdash q$. (Why does this imply what we require, and what choices of x' for x would you make?)

- We seek a model in which $p \rightarrow \Box\Diamond q$ is not true. By definition, we seek a model in which *not every world* satisfies $p \rightarrow \Box\Diamond q$. The model in Figure 5.5 of the textbook qualifies, for $a \not\Vdash p \rightarrow \Box\Diamond q$. (Why?)

5(d). The relation $x \Vdash \Diamond(p \wedge q)$ means that there is a world y with $R(x, y)$ and $y \Vdash p \wedge q$. On the other hand, $x \Vdash \Diamond p \wedge \Diamond q$ means that there are worlds y' and y'' with

$$R(x, y') \text{ and } y' \Vdash p$$
$$R(x, y'') \text{ and } y'' \Vdash q.$$

With that in mind, it is relatively easy to find a model which distinguishes these two formulas:



Note that $a \not\Vdash \Diamond(p \wedge q)$, whereas $a \Vdash \Diamond p \wedge \Diamond q$.

We make use of the second part of Theorem 5.26. Thus, if y is any world which is G-reachable from x within l steps, we have to argue that $y \Vdash C_G\phi$, i.e. $y \Vdash E_G^k\phi$ for all $k \geq 1$. Fix any $k \geq 1$. If z is G-reachable from y within k steps, we are done if $z \Vdash \phi$. But since y is G-reachable from x within l steps, we conclude that z is G-reachable from x within $l + k$ steps. But then our assumption that $x \Vdash C_G\phi$ implies $x \Vdash E_G^{l+k}\phi$ which renders $z \Vdash \phi$. Thus, $C_G\phi \rightarrow C_GC_g\phi$ is valid in KT45.

- To show that $\neg C_G\phi \rightarrow C_G\neg C_G\phi$ is valid in KT45, it suffices to consider an arbitrary world x in an arbitrary model of KT45 such that $x \Vdash \neg C_G\phi$. We then only have to show that $x \Vdash C_G\neg C_G\phi$ follows. We make crucial use of the second part of Theorem 5.26, and apply *proof by contradiction*: Assume that $x \nVdash C_G\neg C_G\phi$. By Theorem 5.26, there exists a world y which is G-reachable by x such that $y \nVdash \neg C_G\phi$, i.e. $y \Vdash C_G\phi$. Since G is non-empty (what does $\neg C_G\phi \rightarrow C_G\neg C_G\phi$ mean if G is empty, and why it is then valid?), we know that the relation

$$\bigcup_{i \in G} R_i \tag{5.1}$$

is an equivalence relation (see the discussion in the textbook after Theorem 5.26). But then x and y are related via this equivalence relation and one satisfies $C_G\phi$, whereas the other one does not. This is a contradiction. (Why?)

10(c). Again, we prove this via showing two separate implications.

(i) We prove $Cp \rightarrow K_iC_p$ by

Cp	ass
CCp	$C\mathbf{4}$ 1
ECp	CE 2
K_iCp	EK_i 3
$Cp \rightarrow K_iCp$	$\rightarrow$i 1–4

(ii) This follows simply by the rule KT.

10(e). The formula scheme $\neg\phi \rightarrow K_i\neg K_i\phi$ means

"If ϕ is false, then agent i knows that he does not know ϕ."