Chapter 13

Logical Properties of Processes

I summed up all systems in a phrase, and all existence in an epigram.

- Oscar Wilde.

Thus far in Part II of the book, we have developed the understanding of what a process is, namely a labelled transition system, as well as the means for describing processes formally with a simple process language. We have also defined when two processes are equivalent – namely when they are bisimilar, which we presented as game equivalent – as well as a procedure for determining if two processes are equivalent.

Determining equivalence between processes is instrumental for finding out if a proposed implementation of a computing system matches its specification. However, we are often not interested in the complete behaviour of a system, but rather only in certain aspects. For example, we may not be interested – for the moment – in what actions a certain system does, but rather we might only want to know if it can ever *deadlock*, that is, evolve into a state in which it can perform no actions. This would be very useful in the analysis of systems which are expected to be perpetual, such as operating systems (particularly those running on critical systems). In other instances we may be interested only in knowing if a given system may or will ever perform a particular action, for example service a request such as printing a document that has been sent to the printer queue.

In this chapter we shall consider a simple logic for expressing properties of systems, as well as the means for determining whether or not a given process satisfies such properties. The properties which the logic can express will be dynamic (behavioural) properties which describe what actions a process can or cannot do, rather than static properties such as how many states a process has which are irrelevant for its correct functioning.

A given property will potentially hold of many different systems, and fail to hold of many others. However, the properties that we express should respect our understanding of equivalence: if a given property holds of a particular process, then it should hold of any other equivalent process. Conversely, if two processes are not equivalent, then you should be able to express a property which distinguishes between these two processes; that is, a property which holds of one of the processes but not the other. The logic which we describe in this chapter is of this nature.

Example 13.1

Consider the following two statements about a particular computer:

1. "The computer consists of three parts: a CPU, a memory unit, and a bus for communicating with the environment."
2. "CONTROL-ALT-DELETE can be pressed; this will shut down the computer, which will then not do anything further."

The first statement does not refer to the (dynamic) behaviour of the computer, but rather to its (static) structure. As such it cannot be used to distinguish between the behaviour of this and any other computer. Another computer may (and likely will) consist of the same three parts yet behave completely differently; while yet another may behave identically to the computer in question despite being built completely differently.

By contrast, the second statement describes one particular aspect of the behaviour of the computer which we may want our computer to demonstrate. Another computer built from the same three parts may be unacceptable if it does not behave the same when you press the CONTROL-ALT-DELETE combination of keystrokes.

13.1 The Mays and Musts of Processes

In trying to understand the differences between the behaviours of the various vending machines in Section 11.4, we were led to making statements such as the following two:

1. *We may do a '*`10p`*' action and end up in a state in which*
   we may do a '`10p`*' action and end up in a state in which*
   we cannot do a '`coffee`*' action.*
2. *We may do a '*`10p`*' action and end up in a state in which*
   no matter how we do a '`10p`*' action*
   we must end up in a state in which
   we cannot do a '`tea`*' action.*

Thus we are expressing capabilities (and inabilities) of a process using the two auxiliary verbs *may* and *must*, which are known as *modal helping verbs* as they help set the modality – necessity or possibility – of the main verb. In fact, we are using these verbs in a very strict manner, namely in the following two contexts:

$\langle a\rangle P$: *we* <u>*may*</u> *do an 'a' action and*
end up in a state in which P is true;

$[a]P$: *no matter how we do an 'a' action*
we <u>*must*</u> *end up in a state in which P is true.*

We will use the above notation, $\langle a\rangle P$ (pronounced "diamond-a" P) and $[a]P$ (pronounced "box-a" P), for writing down such statements.

How, then, can we express the simple property that we may do a 'coffee' action? It doesn't suffice to simply write:

$\langle \texttt{coffee}\rangle$

as – following the translations given above – this reads in English as:

we <u>*may*</u> *do a* '`coffee`' *action and end up in a state in which.*

This is not grammatically correct. In order to complete the sentence, we must indicate a property that we require to be true of the state into which the process evolves after doing the 'coffee' action. Every modality "$\langle a\rangle$" and "$[a]$" has to be followed by some property P.

In this case, however, we don't require anything in particular to be true in the state we get into after doing the 'coffee' action; we are only content that we *can* evolve into such a state. To solve this problem, we can use the property true, which of course is itself true of any process state. Thus, to express the property that we may do a 'coffee' action, we would write:

$\langle \texttt{coffee}\rangle\mathsf{true}$

which more fully says

we <u>*may*</u> *do a* '`coffee`' *action and*
end up in a state in which <u>*true*</u> *is true.*

Although the final clause is redundant, as true is always true (that is, true is true in every state), it is nonetheless necessary in order to turn the expression into a complete logical statement.

We may now express the two properties of our vending machines:

1. $\langle \mathsf{10p}\rangle\langle \mathsf{10p}\rangle\neg\langle \texttt{coffee}\rangle\mathsf{true}$
2. $\langle \mathsf{10p}\rangle[\mathsf{10p}]\neg\langle \texttt{tea}\rangle\mathsf{true}$

If we read these two lines as English statements following the translations given above for the new notation – as well as reading the negation of a property, $\neg P$, as *"it is not the case that P"* – we arrive at the following:

1. *We may do a '*`10p`*' action and end up in a state in which*
   we may do a '`10p`*' action and end up in a state in which*
   it is not the case that
   we may do a '`coffee`*' action and*
   end up in a state in which true is true.
2. *We may do a '*`10p`*' action and end up in a state in which*
   no matter how we do a '`10p`*' action*
   we must end up in a state in which
   it is not the case that
   we may do a '`tea`*' action and*
   end up in a state in which true is true.

Exercise 13.1 (Solution on page 477)

Explain what each of the following properties expresses.

1. $\langle\texttt{coffee}\rangle$true
2. $\langle\texttt{coffee}\rangle$false
3. $[\texttt{coffee}]$true
4. $[\texttt{coffee}]$false

Exercise 13.2 (Solution on page 477)

How can we express the property that we cannot do two 'a' actions in a row? Give your answer using the above notation, and write out your property in English.

Exercise 13.3 (Solution on page 478)

How can we express a property that distinguishes between the clock $\texttt{Cl}$ from Example 11.7 which ticks forever, and the clock $\texttt{Cl}_\star$ from Exercise 11.8 which may tick forever or may stop ticking after any tick? That is, give a property using the above notation which is true of $\texttt{Cl}$ but false of $\texttt{Cl}_\star$. Write out your property in English as well.

13.2 A Modal Logic for Properties

In the previous section we presented the core of a simple logical language for expressing properties which may be true or false of a given process.

In this section we complete the description of this simple logic, which we shall simply call **HML** (for Hennessy-Milner Logic, after its inventors). This language consists essentially of propositional logic with the additional two modal connectives $\langle a\rangle P$ ("diamond-a" P) and $[a]P$ ("box-a" P):

$$P, Q \ ::= \ \mathsf{true} \mid \mathsf{false} \mid \neg P \mid P \wedge Q \mid P \vee Q \mid \langle a\rangle P \mid [a]P.$$

A formula P of **HML** represents a property which may or may not be true in a given state E of a process. If it is true in that state, we shall write $E \models P$ and say that the state E ***satisfies*** the property P; otherwise we will write $E \not\models P$ and say that the state E does *not* satisfy the property P; that is, by $E \not\models P$ we mean $\neg(E \models P)$. If a property is true in *some* state, then we say that the property is ***satisfiable***; and if it is true in *every* state, then we say that it is ***valid***.

Whether or not a property is true in a given state is defined inductively on the structure of the formula P as follows:

- $E \models \mathsf{true}$ for all E.

  The property true is true in *every* state.

- $E \not\models \mathsf{false}$ for all E.

  The property false is *not* true in *any* state.

- $E \models \neg P$ if, and only if, $E \not\models P$.

  The property $\neg P$ is true in a state if, and only if, P is *not* true in that state.

- $E \models P \wedge Q$ if, and only if, $E \models P$ *and* $E \models Q$.

  The property $P \wedge Q$ is true in a state if, and only if, both P and Q are true in that state.

- $E \models P \vee Q$ if, and only if, $E \models P$ *or* $E \models Q$.

  The property $P \vee Q$ is true in a state if, and only if, either P or Q (or both) is true in that state.

- $E \models \langle a\rangle P$ if, and only if, $F \models P$ for *some* state F such that $E \xrightarrow{a} F$.

  The property $\langle a\rangle P$ is true in a state if, and only if, you can do an 'a' transition from that state to a state in which the property P is true.

- $E \models [a]P$ if, and only if, $F \models P$ for *all* F such that $E \xrightarrow{a} F$.

  The property $[a]P$ is true in a state if, and only if, the property P is true in *every* state that you can get to by doing an 'a' transition from that state.

The syntax and semantics of the logic **HML** is summarised in Figure 13.1.

We shall make use of the following shorthand abbreviations:

$E \models \mathsf{true}$	for *all* E.
$E \models \mathsf{false}$	for *no* E.
$E \models \neg P$	if, and only if, $E \not\models P$.
$E \models P \wedge Q$	if, and only if, $E \models P$ *and* $E \models Q$.
$E \models P \vee Q$	if, and only if, $E \models P$ *or* $E \models Q$.
$E \models \langle a \rangle P$	if, and only if, $F \models P$ for *some* F such that $E \xrightarrow{a} F$.
$E \models [a]P$	if, and only if, $F \models P$ for *all* F such that $E \xrightarrow{a} F$.

Figure 13.1: Syntax and semantics of the modal logic **HML**.

$\langle - \rangle P \;=\; \bigvee\limits_{a} \langle a \rangle P$ where the disjunction ranges over the whole set of actions of a process.

This property is true in a state if, and only if, you can do *some* transition from that state to a state in which the property P is true.

$[-]P \;=\; \bigwedge\limits_{a} [a]P$ where the conjunction ranges over the whole set of actions of a process.

This property is true in a state if, and only if, the property P is true in *every* state that you can get to by doing a transition from that state.

$\langle K \rangle P \;=\; \bigvee\limits_{a \in K} \langle a \rangle P$ where K is a set of actions (typically written without set braces, as in $\langle a, b, c \rangle P$).

This property is true in a state if, and only if, you can do an 'a' transition from that state, for some $a \in K$, to a state in which the property P is true. This is the same as $\langle a \rangle P$ when $K = \{a\}$; the same as $\langle - \rangle P$ when K is the set of all actions of a process; and the same as false when $K = \emptyset$.

$[K]P \;=\; \bigwedge\limits_{a \in K} [a]P$ where K is a set of actions (typically written without set braces, as in $[a, b, c]P$).

This property is true in a state if, and only if, the property P is true in *every* state that you can get to by doing an 'a' transition from that state, for some $a \in K$. This is the same as $[a]P$ when $K = \{a\}$; the same as $[-]P$ when K is the set of all actions of a process; and the same as true when $K = \emptyset$.

$\langle -K \rangle P \;=\; \langle \overline{K} \rangle P$ where K is a set of actions (typically written without set braces, as in $\langle -a, b, c \rangle P$).

This property is true in a state if, and only if, you can do an 'a' transition from that state, for some $a \notin K$ (i.e., for some $a \in \overline{K}$), to a state in which the property P is true.

$[-K]P \;=\; [\overline{K}]P$ where K is a set of actions (typically written without set braces, as in $[-a, b, c]P$).

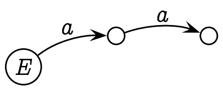
This property is true in a state if, and only if, the property P is true in *every* state that you can get to by doing an 'a' transition from that state, for some $a \notin K$ (i.e., for some $a \in \overline{K}$).

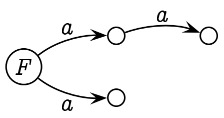
Note that in all of the above shorthand formulæ we assume that the number of possible actions is finite; the logic **HML** does not have infinite conjunction or disjunction.

Example 13.3

Consider the following two simple processes:

$$E \stackrel{\text{def}}{=} a.a.\mathbf{0} \qquad\qquad F \stackrel{\text{def}}{=} a.a.\mathbf{0} \,+\, a.\mathbf{0}$$





These differ in that process F may deadlock immediately after performing the first 'a' action, whereas process E is guaranteed to be able to perform a second 'a' action after performing the first 'a' action. These can be rendered in modal logic as follows:

- $F \models \langle a\rangle\neg\langle a\rangle\mathsf{true}$ whereas $E \not\models \langle a\rangle\neg\langle a\rangle\mathsf{true}$.

  We <u>may</u> do an 'a' action and end up in a state in which we <u>cannot</u> do another 'a' action.

  This is true in state F but not true in state E.

- $E \models [a]\langle a\rangle\mathsf{true}$ whereas $F \not\models [a]\langle a\rangle\mathsf{true}$.

  No matter how we do an 'a' action,
  we <u>must</u> end up in a state in which
  we <u>may</u> do another 'a' action.

  This is true in state E but not true in state F.

When first learning to think logically with the modal verbs "*may*" and "*must*," it is easy to misinterpret properties, particularly when expressing them in the language of **HML**. A common mistake arises when wanting to express the property

I must do an 'a' action.

This property is *not* captured by the formula $\langle a \rangle \mathsf{true}$ which expresses the property

I may do an 'a' action

as this allows the possibility of doing something other than an 'a' action; if, for example, I could also do a 'b' action, then it would clearly not be the case that I *must* do an 'a' action.

The next misconception is that – being a "*must*" property – we would express the desired property (that an 'a' action *must* happen) as $[a]\mathsf{true}$. However, this formula only expresses what must be true *if and when you do an 'a' action*; it doesn't even assert that an 'a' action is even possible! More precisely, it asserts that:

no matter how we do an 'a' action
we must end up in a state in which true is true

which is true of *every* state of a system whether or not it can do an 'a' action!

So how then do we express the property that an 'a' action must happen? The answer is: precisely when an 'a' action *may* happen and *no other action* may happen, which we can express in **HML** as follows:

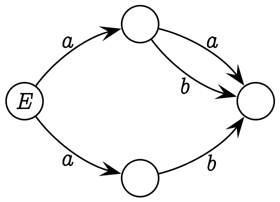
$$\langle a \rangle \mathsf{true} \;\wedge\; \bigwedge_{b \neq a} \neg \langle b \rangle \mathsf{true}$$

or more simply using our shorthand as follows:

$$\langle a \rangle \mathsf{true} \;\wedge\; \neg \langle -a \rangle \mathsf{true}$$

Exercise 13.4 (Solution on page 478)

Consider the following transition system:



Which of the following are correct?

1. $E \models \langle a \rangle \mathsf{true}$	5. $E \models \langle a \rangle \langle a \rangle \mathsf{true}$	9. $E \models [a] \langle a \rangle \mathsf{true}$
2. $E \models \langle b \rangle \mathsf{true}$	6. $E \models \langle a \rangle \langle b \rangle \mathsf{true}$	10. $E \models [a] \langle b \rangle \mathsf{true}$
3. $E \models [a] \mathsf{false}$	7. $E \models \langle a \rangle [a] \mathsf{false}$	11. $E \models [a][a] \mathsf{false}$
4. $E \models [b] \mathsf{false}$	8. $E \models \langle a \rangle [b] \mathsf{false}$	12. $E \models [a][b] \mathsf{false}$

Exercise 13.5 (Solution on page 478)

Express the following properties regarding the lamp process from Section 11.2 pictured in Figure 11.5 (page 289). In each case, indicate which of the three states of the process (OFF, ON, BROKEN) satisfy the property in question.

1. I may do two '*pull*' actions in a row followed by a '*break*' action.
2. I may do two '*pull*' actions in a row followed by a '*reset*' action.
3. I cannot do a '*pull*' action.
4. I can only do a '*pull*' action (that is, I must do a '*pull*' action).

13.3 Negation Is Definable

In Section 11.4 we observed that the negation of a *must* property is equivalent to a *may* property, and vice versa. This should have become apparent as well from Exercise 13.4.

More precisely, we consider two formulæ of our modal logic to be equivalent if, and only if, they are true in the same states: $P \Leftrightarrow Q$ if, and only if, for all states E: $E \models P \;\Leftrightarrow\; E \models Q$. Our observations about negating modal properties are then expressed as follows:

$$\neg\langle a\rangle P \;\Leftrightarrow\; [a]\neg P \qquad \textit{and}$$

$$\neg[a]P \;\Leftrightarrow\; \langle a\rangle\neg P$$

In words these say the following: the property which states that

we <u>cannot</u> do an 'a' action and
end up in a state in which P is true

is equivalent to

no matter how we do an 'a' action
we <u>must</u> end up in a state in which ¬P is true;

and the property which states that

it is <u>not</u> true that no matter how we do an 'a' action
we <u>must</u> end up in a state in which P is true

is equivalent to

we <u>may</u> do an 'a' action and
end up in a state in which ¬P is true.

We can motivate this correspondence by expressing the meaning of the modal connectives in predicate logic. A *may* property says something about *some* state to which you can go, whereas a *must* property says something about *all* states to which you can go:

$$E \models \langle a \rangle P \text{ if, and only if, } \exists F\,(E \xrightarrow{a} F \;\wedge\; F \models P)$$

$$E \models [a]P \text{ if, and only if, } \forall F\,(E \xrightarrow{a} F \;\Rightarrow\; F \models P)$$

We can then reason about these properties using the rules for quantification from Section 4.3:

$$\neg\forall x\, P(x) \;\Leftrightarrow\; \exists x\, \neg P(x) \quad \text{and} \quad \neg\exists x\, P(x) \;\Leftrightarrow\; \forall x\, \neg P(x).$$

For example, we can show the equivalence $\neg\langle a\rangle P \;\Leftrightarrow\; [a]\neg P$ as follows:

$$\begin{aligned} E \models \neg\langle a\rangle P \;&\Leftrightarrow\; E \not\models \langle a\rangle P \\ &\Leftrightarrow\; \neg\exists F\,(E \xrightarrow{a} F \;\wedge\; F \models P) \\ &\Leftrightarrow\; \forall F \neg(E \xrightarrow{a} F \;\wedge\; F \models P) \\ &\Leftrightarrow\; \forall F(E \xrightarrow{a} F \;\Rightarrow\; F \not\models P) \\ &\Leftrightarrow\; \forall F(E \xrightarrow{a} F \;\Rightarrow\; F \models \neg P) \\ &\Leftrightarrow\; E \models [a]\neg P \end{aligned}$$

Exercise 13.6 (Solution on page 478)

Show the equivalence $\neg[a]P \Leftrightarrow \langle a\rangle\neg P$ by using the rules for quantification to prove that $E \models \neg[a]P \;\Leftrightarrow E \models \langle a\rangle\neg P$.

Example 13.6

In order to express that we cannot do an 'a' action, we can write

$\neg\langle a\rangle$true *(It is not the case that we can do an 'a' action.)*

By the above observation, since $\neg$true = false this is equivalent to the expression

$[a]$false *(No matter how we do an 'a' action we must end up in a state in which false is true.)*

Since false cannot be true in *any* state, this means that there must be no possibility of doing an 'a' action, as if we could do an 'a' action we would have to end up in a state in which false is true.

Although **HML** includes negation, we can show that any property that can be expressed in **HML** can be expressed without using negation. That is, any formula P of **HML** can be transformed into a formula $\mathsf{pos}(P)$ which contains no negation symbol and is semantically equivalent to P in the sense that $E \models \mathsf{pos}(P)$ if, and only if, $E \models P$. This transformation is defined together with a dual transformation $\mathsf{neg}(P)$ which transforms the formula P into one which contains no negation symbols yet is semantically equivalent to $\neg P$ in that $E \models \mathsf{neg}(P)$ if, and only if, $E \not\models P$. Both transformations involve pushing negations into formulas using De Morgan's Laws, and are defined inductively on the structure of the formula P as follows:

$$\begin{array}{rclcrcl}
\mathsf{pos}(\mathsf{true}) &=& \mathsf{true} & \qquad & \mathsf{neg}(\mathsf{true}) &=& \mathsf{false} \\
\mathsf{pos}(\mathsf{false}) &=& \mathsf{false} & & \mathsf{neg}(\mathsf{false}) &=& \mathsf{true} \\
\mathsf{pos}(\neg P) &=& \mathsf{neg}(P) & & \mathsf{neg}(\neg P) &=& \mathsf{pos}(P) \\
\mathsf{pos}(P \wedge Q) &=& \mathsf{pos}(P) \wedge \mathsf{pos}(Q) & & \mathsf{neg}(P \wedge Q) &=& \mathsf{neg}(P) \vee \mathsf{neg}(Q) \\
\mathsf{pos}(P \vee Q) &=& \mathsf{pos}(P) \vee \mathsf{pos}(Q) & & \mathsf{neg}(P \vee Q) &=& \mathsf{neg}(P) \wedge \mathsf{neg}(Q) \\
\mathsf{pos}(\langle a\rangle P) &=& \langle a\rangle\mathsf{pos}(P) & & \mathsf{neg}(\langle a\rangle P) &=& [a]\mathsf{neg}(P) \\
\mathsf{pos}([a]P) &=& [a]\mathsf{pos}(P) & & \mathsf{neg}([a]P) &=& \langle a\rangle\mathsf{neg}(P)
\end{array}$$

It is immediately clear that $\mathsf{pos}(P)$ and $\mathsf{neg}(P)$ are negation-free terms, as negation does not appear on the right-hand side of any of the defining equations.

Theorem 13.6

For any process E and any formula P of **HML**:

1. $E \models \mathsf{pos}(P)$ *if, and only if,* $E \models P$; *and*
2. $E \models \mathsf{neg}(P)$ *if, and only if,* $E \not\models P$.

Proof: By induction on the structure of P. The details are left as an exercise.

Exercise 13.7 (Solution on page 478)

Prove Theorem 13.6

Exercise 13.8 (Solution on page 482)

Prove, by induction on the structure of P, that $\mathsf{neg}(\mathsf{neg}(P)) = P$.

13.4 The Vending Machines Revisited

We can now express precisely the differences between the three vending machines V_1, V_2 and V_3 introduced in Section 11.4, by writing down formulæ of the logic **HML** which distinguish between them. Specifically, we shall produce three formulæ P_1, P_2 and P_3 of **HML** such that $V_i \models P_i$ for each i, but $V_i \not\models P_j$ whenever $i \neq j$. That is, formula P_i will distinguish the machine V_i from the other two machines by expressing a property which is true of machine V_i but not true of the others.

1. $P_1 \;=\; [\texttt{10p}][\texttt{10p}]\langle\texttt{tea}\rangle\mathsf{true}$

   This formula expresses the property that after doing two consecutive '10p' actions, we must be in a state in which we can do a `tea` move. This is true of V_1 as there is only one state in which we can be after doing the two '10p' actions, namely the state

   `coffee.collect.`V_1 `+ tea.collect.`V_1

   and it is certainly the case that we may do a `tea` move from this state.

   However, this is neither true of V_2 nor of V_3; in both of these cases it is possible to do two consecutive '10p' actions and end up in a state where a '`tea`' action is *not* possible (only a '`coffee`' action). That is, V_2 and V_3 satisfy the formula

   $P_1' \;=\; \langle\texttt{10p}\rangle\langle\texttt{10p}\rangle[\texttt{tea}]\mathsf{false}$

   while V_1 does not. (This formula is the negation of the one in question.)

2. $P_2 \;=\; [\texttt{10p}]\langle\texttt{10p}\rangle[\texttt{tea}]\mathsf{false}$

   This formula expresses the property that after doing a '10p' action, we will be able to do a further '10p' action and end up in a state where we cannot do a '`tea`' action. This is true of V_2 as there is only one state in which we can be after doing the first '10p' action, namely the state

   `10p.coffee.collect.`V_1 `+ 10p.tea.collect.`V_1

   and we can indeed do a further '10p' action, getting to the state

   `coffee.collect.`V_1

   in which we cannot do a '`tea`' action.

   However, this is neither true of V_1 nor of V_3; in these cases it is possible to do the following '10p' actions:

   - $V_1 \xrightarrow{\texttt{10p}}$ `10p.(coffee.collect.`V_1 `+ tea.collect.`V_1`)`

- $V_3 \xrightarrow{\texttt{10p}}$ `10p.tea.collect.`V_3

In both cases we end up in a state from which, after doing a further '`10p`' action, we can do a '`tea`' action. V_1 and V_3 thus satisfy the formula

$$P_2' = \langle\texttt{10p}\rangle[\texttt{10p}]\langle\texttt{tea}\rangle\mathsf{true}$$

while V_2 does not. (This formula is the negation of the one in question.)

3. $P_3 = \langle\texttt{10p}\rangle[\texttt{10p}][\texttt{tea}]\mathsf{false}$

   This formula expresses the property that it is possible to do a '`10p`' action and end up in a state from which we cannot do a further '`10p`' action followed by a '`tea`' action. This is true of V_3 as we can make the transition

   $$V_3 \xrightarrow{\texttt{10p}} \texttt{10p.coffee.collect.}V_3$$

   and indeed find ourselves in a state from which we cannot do a further '`10p`' action followed by a '`tea`' action.

   However, this is neither true of V_1 nor of V_2; in each of these cases there is only one `10p` transition possible, namely:

   - $V_1 \xrightarrow{\texttt{10p}}$ `10p.(coffee.collect.`V_1 `+ tea.collect.`V_1`)`
   - $V_2 \xrightarrow{\texttt{10p}}$ `10p.coffee.collect.`V_2 `+ 10p.tea.collect.`V_2

   In both cases we end up in a state from which we can do a further '`10p`' action followed by a '`tea`' action. V_1 and V_2 thus satisfy the formula

   $$P_3' = [\texttt{10p}]\langle\texttt{10p}\rangle\langle\texttt{tea}\rangle\mathsf{true}$$

   while V_3 does not. (This formula is the negation of the one in question.)

Exercise 13.9 (Solution on page 482)

Recall the following processes from Exercise 11.16.

$$A \overset{\text{def}}{=} b.c.\mathbf{0} + b.d.\mathbf{0} \qquad C \overset{\text{def}}{=} a.B + a.A$$
$$B \overset{\text{def}}{=} A + b.(c.\mathbf{0} + d.\mathbf{0}) \qquad D \overset{\text{def}}{=} a.B$$

Give two formulæ of **HML** which distinguish between C and D: one formula which is true of C but not true of D; and one formula which is true of D but not true of C.

13.5 Modal Properties and Bisimulation

We have now developed two methods for distinguishing between processes.

1. In Chapter 12 we explicitly defined what it means for two processes to be equivalent, in terms of winning strategies in games.
2. In this chapter we defined a modal logic for expressing properties of processes with which we can distinguish between two processes.

We may well wonder if these two methods give the same results.

1. We should be disturbed if two equivalent processes could be differentiated by some formula of the modal logic. This would question the usefulness of the logic as a tool for reasoning about the behaviour of processes.
2. It would also be disappointing, though less of a concern, if the modal logic could not distinguish between some pair of non-equivalent processes. This would mean simply that the logic is too weak to express all aspects of the behaviour of a process.

However, we devised the equivalence based on a consideration of the capabilities of the processes as expressed using precisely the types of modal verbs which form the basis of our logic **HML**. Hence our intuition suggests that the distinguishing power of the modal logic should coincide with the equivalence. In this section we explore and confirm this intuition.

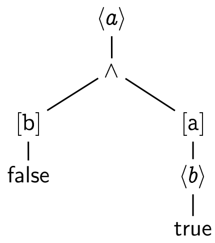
To determine if two processes are n-game equivalent, we need to explore only the first n transitions of the processes; the behaviour of the processes after n transitions is irrelevant. In the same way, in order to determine whether or not some formula of the modal logic is true of some process, we need only to explore the initial behaviour of the process; exactly how deeply we need explore the process depends on the complexity of the formula, as defined by its modal depth.

Definition 13.9

The ***modal depth*** $md\,(P)$ *of a formula* P *of* **HML** *is defined inductively as follows.*

$$\begin{aligned} md\,(\mathit{true}) &= 0 & \qquad md\,(P \wedge Q) &= \max(md\,(P)\,,\, md\,(Q)) \\ md\,(\mathit{false}) &= 0 & md\,(P \vee Q) &= \max(md\,(P)\,,\, md\,(Q)) \\ md\,(\neg P) &= md\,(P) & md\,(\langle a\rangle P) &= 1 + md\,(P) \\ & & md\,([a]P) &= 1 + md\,(P) \end{aligned}$$

The modal depth simply counts the maximum number of modal operators along any path in the syntax tree of an **HML** formula. For example, the formula $\langle a\rangle([b]\mathsf{false} \wedge [a]\langle b\rangle\mathsf{true})$ has a modal depth of 3, as evidenced by the following syntax tree for the formula.



$$\begin{aligned}
&md\,(\langle a\rangle([b]\mathsf{false} \wedge [a]\langle b\rangle\mathsf{true}))\\
&\quad= 1 + md\,([b]\mathsf{false} \wedge [a]\langle b\rangle\mathsf{true})\\
&\quad= 1 + \max(md\,([b]\mathsf{false}),\\
&\qquad\qquad md\,([a]\langle b\rangle\mathsf{true}))\\
&\quad= 1 + \max(1,2)\\
&\quad= 1 + 2\\
&\quad= 3
\end{aligned}$$

The following theorem demonstrates that no formula of modal depth n can distinguish between two processes which are n-game equivalent. The immediate corollary to this is our first desired result: that we cannot use the logic to distinguish between equivalent processes.

Theorem 13.9

If $E \models P$ and $E \sim_n F$ where $n = md\,(P)$, then $F \models P$. That is, no formula of modal depth n can distinguish between two n-game equivalent processes.

Proof: By induction on the structure of P, arguing by cases on the structure of P:

$\underline{P = \mathbf{true}}$: The result is immediately true in this case, as the conclusion $F \models \mathsf{true}$ is always true.

$\underline{P = \mathbf{false}}$: The result is vacuously true in this case, as the premise $E \models \mathsf{false}$ is false.

$\underline{P = \neg Q}$: Since $E \models \neg Q$, we have $E \not\models Q$, and hence $F \not\models Q$ by induction, so $F \models \neg Q$.

$\underline{P = Q_1 \wedge Q_2}$: Note that $n_1 = md\,(Q_1) \leq n$ and $n_2 = md\,(Q_2) \leq n$; hence $E \sim_{n_1} F$ and $E \sim_{n_2} F$.

Since $E \models Q_1 \wedge Q_2$, we have that $E \models Q_1$ and that $E \models Q_2$. Hence by the induction hypothesis (applied twice), we have that $F \models Q_1$ and that $F \models Q_2$, and thus that $F \models Q_1 \wedge Q_2$.

$\underline{P = Q_1 \vee Q_2}$: Note that $n_1 = md\,(Q_1) \leq n$ and $n_2 = md\,(Q_2) \leq n$; hence $E \sim_{n_1} F$ and $E \sim_{n_2} F$.

Since $E \models Q_1 \vee Q_2$, we have that $E \models Q_1$ or that $E \models Q_2$. Hence by the induction hypothesis (applied twice), we have that $F \models Q_1$ or that $F \models Q_2$, and thus that $F \models Q_1 \vee Q_2$.

<u>$P = \langle a \rangle Q$</u>: Note first that $n = md(P) > 0$, and $md(Q) = n-1$.

Since $E \models \langle a \rangle Q$, we have that $E \xrightarrow{a} E'$ for some E' such that $E' \models Q$. But then, since $E \sim_n F$, we must have that $F \xrightarrow{a} F'$ for some F' such that $E' \sim_{n-1} F'$. Hence by the induction hypothesis, we have that $F' \models Q$, and thus that $F \models \langle a \rangle Q$ as required.

<u>$P = [a]Q$</u>: Note first that $n = md(P) > 0$, and $md(Q) = n-1$.

To show that $F \models [a]Q$, we need to show that $F' \models Q$ whenever $F \xrightarrow{a} F'$.

Suppose then that $F \xrightarrow{a} F'$. Since $E \sim_n F$ we must have that $E \xrightarrow{a} E'$ for some E' such that $E' \sim_{n-1} F'$; furthermore, since $E \models [a]Q$, we must have that $E' \models Q$. Thus by the induction hypothesis, we must have that $F' \models Q$ as required. $\square$

We can express this result more succinctly if we first formulate the notion of logical equivalence with respect to the formulæ of **HML** of a fixed bounded modal depth.

Definition 13.10

Let

$$\mathbf{HML}_n = \{ P \in \mathbf{HML} : md(P) \leq n \}$$

be the subset of **HML** *consisting of all formulæ of modal depth at most n.*

1. *Two processes E and F are **n-logically equivalent**, written $E \equiv_n F$, if, and only if, the following holds.*

   For all $P \in \mathbf{HML}_n$: $E \models P$ if, and only if, $F \models P$.

   That is, no formula of modal depth n (or less) can distinguish between them.

2. *The processes E and F are **logically equivalent**, written $E \equiv F$ if, and only if, the following holds.*

   for all $P \in \mathbf{HML}$: $E \models P$ if, and only if, $F \models P$.

   That is, no formula (of any modal depth) can distinguish between them.

Theorem 13.9 then states simply that $E \equiv_n F$ whenever $E \sim_n F$.

Corollary 13.10

If $E \sim F$ then $E \equiv F$, that is, no formula of the logic **HML** *can distinguish between two equivalent processes E and F.*

Proof: If E and F *could* be differentiated by a formula P of **HML**, then by the above we would have that $E \not\sim_n F$, where $n = md\,(P)$, and hence we would have that $E \not\sim F$. □

The converse result, that two processes which cannot be distinguished by any property of **HML** must be equivalent, is not completely attainable. This is due to the fact that equivalence is not the limit of the n-game equivalences, while the logic **HML** is the limit of the bounded logics $\mathbf{HML}_n$. However, as was the case with relating the finite-game equivalences to bisimulation equivalence, this result holds when we restrict ourselves to image-finite processes.

Theorem 13.10

For image-finite processes E and F, if $E \equiv_n F$ then $E \sim_n F$.

Proof: We shall prove, by induction on n, the equivalent contrapositive statement that if $E \not\sim_n F$ then there is a formula P of modal depth n such that $E \models P$ but $F \not\models P$.

The base case (n=0) is vacuously true, as the premise $E \not\sim_0 F$ cannot hold.

For the induction step, suppose that $E \not\sim_{n+1} F$, and assume (without loss of generality) that $E \xrightarrow{a} E'$ for some E' such that $E' \not\sim_n F'$ whenever $F \xrightarrow{a} F'$. Let

$$F \xrightarrow{a} F_1 \qquad F \xrightarrow{a} F_2 \qquad \cdots \qquad F \xrightarrow{a} F_k$$

be all of the (finitely-many) a-transitions possible from F. Then $E' \not\sim_n F_i$ for each $i = 1, 2, \ldots, k$, and hence by the induction hypothesis there are properties $P_1, P_2, \ldots, P_k$ of modal depth n such that $E' \models P_i$ but $F_i \not\models P_i$ for each $i = 1, 2, \ldots, k$. The property P we seek is then

$$P \;=\; \langle a\rangle(P_1 \wedge P_2 \wedge \cdots \wedge P_k).$$

Clearly $E \models \langle a\rangle(P_1 \wedge P_2 \wedge \cdots \wedge P_k)$ but $F \not\models \langle a\rangle(P_1 \wedge P_2 \wedge \cdots \wedge P_k)$. □

Corollary 13.11

For image-finite processes E and F, if $E \equiv F$ then $E \sim F$.

Proof: If $E \equiv F$ then $E \equiv_n F$ for all n, and hence by the above, $E \sim_n F$ for all n. Thus, since E and F are image-finite, $E \sim F$. $\square$

The clock processes $\texttt{Clock}$ and $\texttt{Clock}_\star$ from Example 11.11 pictured in Figure 11.7 (page 298) provide the counter-example to this Corollary in the case of infinite-branching processes. In this case, $\texttt{Clock} \sim_n \texttt{Clock}_\star$ for all $n \in \mathbb{N}$, and hence $\texttt{Clock} \equiv_n \texttt{Clock}_\star$ for all $n \in \mathbb{N}$, meaning that $\texttt{Clock} \equiv \texttt{Clock}_\star$; however, $\texttt{Clock} \not\sim \texttt{Clock}_\star$.

★ 13.6 Characteristic Formulæ

Given a process state E, a formula $cf(E)$ of the logic **HML** is called a ***characteristic formula*** for E if, and only if, for all processes F:

$$F \models cf(E) \text{ if, and only if, } F \sim E.$$

For example, the characteristic formula for $\mathbf{0}$ is

$$cf(\mathbf{0}) = [-]\mathsf{false}$$

as this formula specifies that there are no transitions possible from the state in question.

Exercise 13.11 (Solution on page 483)

1. Argue that the characteristic formula for $a.\mathbf{0}$ is

$$cf(a.\mathbf{0}) = \langle a\rangle\mathsf{true} \wedge [-a]\mathsf{false} \wedge [-][-]\mathsf{false}.$$

2. Give a characteristic formula for $a.(b.\mathbf{0} + c.\mathbf{0})$.

The existence of characteristic formulæ further cements the close connection between modal properties and bisimilarity. However, the exact relationship as presented in the following theorem takes into account the finite limitation of modal formulæ: that they can reason only about the first steps of a process up to a number of steps equal to the modal depth of the formula.

Theorem 13.11

For every $n \in \mathbb{N}$ and every state E of an LTS defined over a finite set of actions, there is a formula $cf_n(E) \in \mathbf{HML}_n$ such that, for all states F,

$$F \models cf_n(E) \quad \textit{iff} \quad E \sim_n F.$$

Furthermore, for every $n \in \mathbb{N}$ there are only finitely-many such formulæ $cf_n(E)$.

Proof: By induction on n.

For the base case we can take $cf_n(E) = \mathsf{true}$, since $F \models \mathsf{true}$ and $E \sim_0 F$ for every $F \in$ States. Clearly there are only finitely-many (namely, one) such formulæ.

For the induction step, let

$$\begin{aligned} cf_{n+1}(E) \;=\; & \bigwedge \left\{ \langle a \rangle\, cf_n(E') \,:\, E \xrightarrow{a} E' \right\} \\ & \wedge \bigwedge_{a \in A} [a] \bigvee \left\{ cf_n(E') \,:\, E \xrightarrow{a} E' \right\}. \end{aligned}$$

There are two parts to this formula:

- The first conjunction of subformulæ characterises what transition are possible: for each transition $E \xrightarrow{a} E'$, it must be possible to do an a transition into a state characterised by the formula $cf_n(E')$.
- The second conjunction of subformulæ characterises the states into which such a transition must evolve: upon performing an a transition, the process must evolve into a state characterised by the formula $cf_n(E')$ for some E' such that $E \xrightarrow{a} E'$.

Recalling the assumption that A is finite, we can note that – even though there may be infinitely-many transitions $E \xrightarrow{a} E'$ – the two sets of subformulæ are, by induction, finite; hence, this is a well-formed formula (ie, it is of finite size), and there can only be finitely-many such formulæ.

Suppose that $F \models cf_{n+1}(E)$.

- If $E \xrightarrow{a} E'$ then, since $F \models \langle a \rangle\, cf_n(E')$,

  $F \xrightarrow{a} F'$ for some F' such that $F' \models cf_n(E')$,

  and thus by induction $E' \sim_n F'$.
- If $F \xrightarrow{a} F'$ then, since $F \models [a] \bigvee_{E \xrightarrow{a} E'} cf_n(E')$,

  $E \xrightarrow{a} E'$ for some E' such that $F' \models cf_n(E')$,

  and thus by induction $E' \sim_n F'$.

Hence, by the above Lemma, $E \sim_{n+1} F$.

Now suppose that $E \sim_{n+1} F$.

- If $E \xrightarrow{a} E'$ then, by the above Lemma, $F \xrightarrow{a} F'$ for some F' such that $E' \sim_n F'$, and thus by induction $F' \models cf_n(E')$. As this is true of all $a \in A$ and all E' such that $E \xrightarrow{a} E'$,

$$\begin{aligned}
\|\mathsf{true}\| &= \mathsf{States}\\
\|\mathsf{false}\| &= \emptyset\\
\|\neg P\| &= \overline{\|P\|}\\
\|P \wedge Q\| &= \|P\| \cap \|Q\|\\
\|P \vee Q\| &= \|P\| \cup \|Q\|\\
\|\langle a\rangle P\| &= \{\, E \in \mathsf{States} \ :\ E \xrightarrow{a} E' \text{ for some } E' \in \|P\|\,\}\\
\|[a]P\| &= \{\, E \in \mathsf{States} \ :\ E \xrightarrow{a} E' \text{ implies } E' \in \|P\|\,\}
\end{aligned}$$

Figure 13.2: Global semantics of the modal logic **HML**.

$$F \models \bigwedge_{E \xrightarrow{a} E'} \langle a\rangle\, cf_n(E')$$

- If $F \xrightarrow{a} F'$ then, by the above Lemma, $E \xrightarrow{a} E'$ for some E' such that $E' \sim_n F'$, and thus by induction $F' \models cf_n(E')$. As this is true of all $a \in A$ and all F' such that $F \xrightarrow{a} F'$,

$$F \models \bigwedge_{a\in A} [a] \bigvee_{E \xrightarrow{a} E'} cf_n(E')$$

Hence $F \models cf_{n+1}(E)$. □

⋆ 13.7 Global Semantics

An alternative way to define the semantics of properties of **HML** is by associating to each property $P \in$ **HML** the set $\|P\|$ of states which satisfy the property P. Determining whether or not $E \models P$ then would correspond to determining if $E \in \|P\|$.

An inductive definition of the semantic function $\|P\|$ is given in Figure 13.2, where the set States represents the set of all states of the underlying transition system. With this definition, we get the following result.

Theorem 13.12

$E \models P$ if, and only if, $E \in \|P\|$.

Proof: By induction on the structure of P, arguing by cases on the structure of P.

<u>$P =$ **true**</u>: $E \models \mathsf{true} \ \Leftrightarrow\ E \in \mathsf{States} \ \Leftrightarrow\ E \in \|\mathsf{true}\|$

$\underline{P = \textbf{false}}$: $E \models \text{false} \;\Leftrightarrow\; E \in \emptyset \;\Leftrightarrow\; E \in ||\text{false}||$

$\underline{P = \neg P}$: $E \models \neg P \;\Leftrightarrow\; E \not\models P \;\Leftrightarrow\; E \notin ||P|| \;\Leftrightarrow\; E \in \overline{||P||} \;\Leftrightarrow\; E \in ||\neg P||$

$\underline{P = Q_1 \wedge Q_2}$: $E \models Q_1 \wedge Q_2 \Leftrightarrow\; E \models Q_1$ and $E \models Q_2$

$$\Leftrightarrow\; E \in ||Q_1|| \cap ||Q_2|| \quad\Leftrightarrow\quad E \in ||Q_1 \wedge Q_2||$$

$\underline{P = Q_1 \vee Q_2}$: $E \models Q_1 \vee Q_2 \Leftrightarrow\; E \models Q_1$ or $E \models Q_2$

$$\Leftrightarrow\; E \in ||Q_1|| \cup ||Q_2|| \quad\Leftrightarrow\quad E \in ||Q_1 \vee Q_2||$$

$\underline{P = \langle a \rangle Q}$: $E \models \langle a \rangle Q \Leftrightarrow\; E \xrightarrow{a} E'$ such that $E' \models Q$

$$\Leftrightarrow\; E \xrightarrow{a} E' \text{ such that } E' \in ||Q|| \quad\Leftrightarrow\quad E \in ||\langle a \rangle Q||$$

$\underline{P = [a]Q}$: $E \models [a]Q \Leftrightarrow\; E \xrightarrow{a} E'$ implies $E' \models Q$

$$\Leftrightarrow\; E \xrightarrow{a} E' \text{ implies } E' \in ||Q|| \quad\Leftrightarrow\quad E \in ||[a]Q|| \qquad \square$$

Exercise 13.12 (Solution on page 483)

Consider the following transition system:

$$\begin{tikzcd}
 & E_1 \arrow[dr, bend left, "a"] \arrow[dr, "b"'] & \\
E \arrow[ur, bend left, "a"] \arrow[dr, bend right, "a"'] & & F \arrow[dl, bend right, "a"'] \\
 & E_2 \arrow[ur, bend right, "b"'] &
\end{tikzcd}$$

Compute the following sets:

1. $||\langle a \rangle \text{true}||$
2. $||\langle b \rangle \text{true}||$
3. $||\langle a \rangle \langle a \rangle \text{true}||$
4. $||\langle b \rangle \langle b \rangle \text{true}||$
5. $||\langle a \rangle [a] \text{false}||$
6. $||[b] \langle a \rangle \text{true}||$

13.8 Additional Exercises

1. Give properties of the modal logic **HML** which distinguish between the clocks $\texttt{Cl}_n$ of Example 11.7. That is, for each $n \in \mathbb{N}$, give a formula P_n of **HML** which is true of $\texttt{Cl}_n$ but false of $\texttt{Cl}_k$ for every $k \neq n$.
2. What does the property $\langle a \rangle$false say? Can you give an example process which satisfies this property?

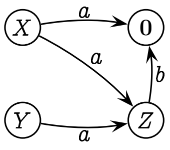
3. Express the negation of each of the following properties without using negation operator $\neg$. In each case, write out each property and its negation in English.

   (a) $[a](\langle b\rangle\mathsf{true} \wedge \langle c\rangle\mathsf{true})$.

   (b) $[a]\langle b\rangle(\langle a\rangle\mathsf{true} \vee \langle b\rangle[a]\mathsf{false})$.

4. Consider the following 4-state transition system.

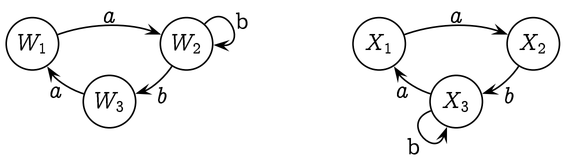
$$X \overset{\text{def}}{=} a.\mathbf{0} + Y$$
$$Y \overset{\text{def}}{=} a.Z$$
$$Z \overset{\text{def}}{=} b.\mathbf{0}$$



Fill in the following table with the states satisfying the relevant properties. (The first line has been filled in to get you started.)

property P	states satisfying P	negation $\neg P$	states satisfying $\neg P$
$\langle a\rangle\mathsf{true}$	X, Y	$[a]\mathsf{false}$	Z, $\mathbf{0}$
$[a]\mathsf{true}$			
$\langle b\rangle\mathsf{true}$			
$[b]\mathsf{true}$			
$\langle a\rangle\langle b\rangle\mathsf{true}$			
$\langle a\rangle[b]\mathsf{true}$			
$[a]\langle b\rangle\mathsf{true}$			
$[a][b]\mathsf{true}$			

5. Consider the following labelled transition system.



Give a modal logic formula which distinguishes between W_1 and X_1. Argue why no formula of smaller modal depth can distinguish between these two states.

6. Give a labelled transition system with a state s which satisfies all of the following:

   - $\langle a\rangle(\langle a\rangle\mathsf{true} \ \wedge\ \langle b\rangle\langle a\rangle\mathsf{true})$
   - $\langle a\rangle\langle b\rangle(\langle b\rangle\mathsf{true} \ \wedge\ [a]\mathsf{false})$
   - $\langle a\rangle\langle b\rangle([a]\mathsf{false} \ \wedge\ [b]\mathsf{false})$

7. Recall the specification of the car safety system from Exercise 11 on page 305.

   (a) Express $R(x)$ in the modal logic M in two ways:

      i. one way involving only the action "ring"; and
      ii. another way *not* involving the action "ring".

      (Hint: First express $R(x)$ in terms of $D(x)$, $B(x)$ and $M(x)$.)

   (b) Which states satisfy the following formulæ?

      i. $\langle \text{buckle}\rangle\mathsf{true} \ \wedge\ \langle \text{close}\rangle\mathsf{true}$
      ii. $\langle \text{buckle}\rangle\mathsf{true} \ \wedge\ [\text{close}]\mathsf{false}$
      iii. $\langle \text{on}\rangle\langle \text{ring}\rangle\mathsf{true}$
      iv. $[\text{on}]\langle \text{ring}\rangle\mathsf{true}$
      v. $\langle \text{open}\rangle\big(\langle \text{buckle}\rangle\mathsf{true} \ \wedge\ \langle \text{off}\rangle\mathsf{true}\big)$
      vi. $\langle \text{open}\rangle\big(\langle \text{buckle}\rangle\mathsf{true} \ \vee\ \langle \text{off}\rangle\mathsf{true}\big)$

8. Prove or disprove the following. (Here, equality between formulæ means that the formulæ express the same properties.)

   (a) $\langle a\rangle(P \wedge Q) \ = \ \langle a\rangle P \wedge \langle a\rangle Q.$
   (b) $\langle a\rangle(P \vee Q) \ = \ \langle a\rangle P \vee \langle a\rangle Q.$
   (c) $[a](P \wedge Q) \ = \ [a]P \wedge [a]Q$
   (d) $[a](P \vee Q) \ = \ [a]P \vee [a]Q.$

9. As defined, the modal logic **HML** involves only binary conjunctions and disjunctions, $P \wedge Q$ and $P \vee Q$, and thus by extension finite conjunctions and disjunctions, $\bigwedge \mathcal{F}$ and $\bigvee \mathcal{F}$ for finite sets of formulæ $\mathcal{F}$.

   Prove that if we allow infinite conjunctions and disjunctions, then the logical characterisation of bisimulation equivalence is tight: that $E \sim F$ if, and only if, E and F satisfy the same (infinitary) modal logic formulæ.

10. Let $\mathbf{HML}_t$ be the subset of **HML** formulæ generated by the following BNF equation:

$$P \ ::= \ \mathsf{true} \ \mid \ \langle a\rangle P$$

   Show that $\mathbf{HML}_t$ characterises trace equivalence $=_t$ from Exercise 8 on page 330, in the sense that $E =_t F$ if, and only if, E and F satisfy the same formulæ of $\mathbf{HML}_t$.

11. Let $\mathsf{\mathbf{HML}}_{\asymp}$ be the subset of **HML** formulæ generated by the following BNF equation:

$$P, Q \quad ::= \quad \mathsf{true} \quad | \quad \langle a \rangle P \quad | \quad P \wedge Q$$

Show that $\mathsf{\mathbf{HML}}_{\asymp}$ characterises simulation equivalence $\asymp$ from Exercise 12.8 on page 318, in the sense that $E \asymp F$ if, and only if, E and F satisfy the same formulæ of $\mathsf{\mathbf{HML}}_{\asymp}$.

12. Let $\mathsf{\mathbf{HML}}_f$ be the subset of **HML** formulæ generated by the following BNF equation:

$$P \quad ::= \quad [K]\mathsf{false} \quad | \quad \langle a \rangle P$$

where $K \subseteq \Sigma$ is a set of actions. Show that $\mathsf{\mathbf{HML}}_f$ characterises failures equivalence $=_f$ from Exercise 10 on page 331, in the sense that $E =_f F$ if, and only if, E and F satisfy the same formulæ of $\mathsf{\mathbf{HML}}_f$.