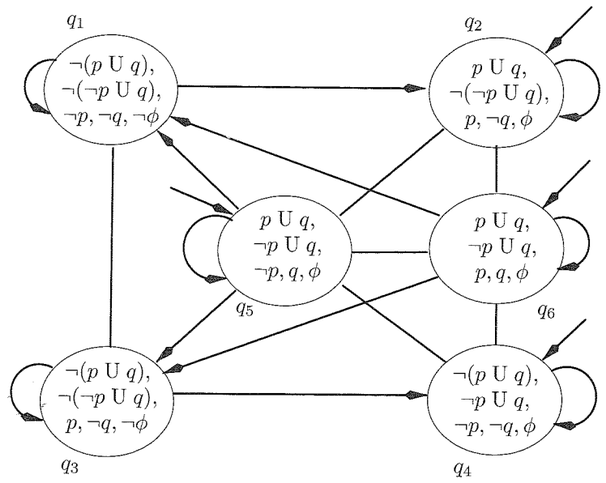

Figure 3.36. Automaton accepting precisely traces satisfying $\phi \stackrel{\text{def}}{=} (p \mathrm{U} q) \vee (\neg p \mathrm{U} q)$. The transitions with no arrows can be taken in either direction. The acceptance condition asserts that every run must pass infinitely often through the set $\{q_1, q_3, q_4, q_5, q_6\}$, and also the set $\{q_1, q_2, q_3, q_5, q_6\}$.

and checking whether there is a path of the resulting system which satisfies the acceptance condition of $A_{\neg\phi}$.

It is possible to implement the check for such a path in terms of CTL model checking, and this is in fact what NuSMV does. The combined system $\mathcal{M} \times A_{\neg\phi}$ is represented as the system to be model checked in NuSMV, and the formula to be checked is simply $\mathrm{EG}\,\top$. Thus, we ask the question: does the combined system have a path. The acceptance conditions of $A_{\neg\phi}$ are represented as implicit fairness conditions for the CTL model-checking procedure. Explicitly, this amounts to asserting 'FAIRNESS $\neg(\chi \mathrm{U} \psi) \vee \psi$' for each formula $\chi \mathrm{U} \psi$ occurring in $\mathcal{C}(\phi)$.

3.7 The fixed-point characterisation of CTL

On page 227, we presented an algorithm which, given a CTL formula ϕ and a model $\mathcal{M} = (S, \rightarrow, L)$, computes the set of states $s \in S$ satisfying ϕ. We write this set as $[\![\phi]\!]$. The algorithm works recursively on the structure of ϕ. For formulas ϕ of height 1 ($\bot$, $\top$ or p), $[\![\phi]\!]$ is computed directly. Other

formulas are composed of smaller subformulas combined by a connective of CTL. For example, if ϕ is $\psi_1 \vee \psi_2$, then the algorithm computes the sets $[\![\psi_1]\!]$ and $[\![\psi_2]\!]$ and combines them in a certain way (in this case, by taking the union) in order to obtain $[\![\psi_1 \vee \psi_2]\!]$.

The more interesting cases arise when we deal with a formula such as $\mathrm{EX}\,\psi$, involving a temporal operator. The algorithm computes the set $[\![\psi]\!]$ and then computes the set of all states which have a transition to a state in $[\![\psi]\!]$. This is in accord with the semantics of $\mathrm{EX}\,\psi$: $\mathcal{M}, s \models \mathrm{EX}\,\psi$ iff there is a state s' with $s \rightarrow s'$ and $\mathcal{M}, s' \models \psi$.

For most of these logical operators, we may easily continue this discussion to see that the algorithms work just as expected. However, the cases EU, AF and EG (where we needed to iterate a certain labelling policy until it stabilised) are not so obvious to reason about. The topic of this section is to develop the semantic insights into these operators that allow us to provide a complete proof for their termination and correctness. Inspecting the pseudo-code in Figure 3.28, we see that most of these clauses just do the obvious and correct thing according to the semantics of CTL. For example, try out what `SAT` does when you call it with $\phi_1 \rightarrow \phi_2$.

Our aim in this section is to prove the termination and correctness of $\mathtt{SAT}_{\mathtt{AF}}$ and $\mathtt{SAT}_{\mathtt{EU}}$. In fact, we will also write a procedure $\mathtt{SAT}_{\mathtt{EG}}$ and prove its termination and correctness[1]. The procedure $\mathtt{SAT}_{\mathtt{EG}}$ is given in Figure 3.37 and is based on the intuitions given in Section 3.6.1: note how *deleting* the label if none of the successor states is labelled is coded as *intersecting* the labelled set with the set of states which have a labelled successor.

The semantics of $\mathrm{EG}\,\phi$ says that $s_0 \models \mathrm{EG}\,\phi$ holds iff there exists a computation path $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \ldots$ such that $s_i \models \phi$ holds *for all* $i \geq 0$. We could instead express it as follows: $\mathrm{EG}\,\phi$ holds if ϕ holds and $\mathrm{EG}\,\phi$ holds in one of the successor states to the current state. This suggests the equivalence $\mathrm{EG}\,\phi \equiv \phi \wedge \mathrm{EX}\,\mathrm{EG}\,\phi$ which can easily be proved from the semantic definitions of the connectives.

Observing that $[\![\mathrm{EX}\,\psi]\!] = \mathrm{pre}_\exists([\![\psi]\!])$ we see that the equivalence above can be written as $[\![\mathrm{EG}\,\phi]\!] = [\![\phi]\!] \cap \mathrm{pre}_\exists([\![\mathrm{EG}\,\phi]\!])$. This does not look like a very promising way of calculating $\mathrm{EG}\,\phi$, because we need to know $\mathrm{EG}\,\phi$ in order to work out the right-hand side. Fortunately, there is a way around this apparent circularity, known as computing fixed points, and that is the subject of this section.

[1] Section 3.6.1 handles $\mathrm{EG}\,\phi$ by translating it into $\neg\mathrm{AF}\,\neg\phi$, but we already noted in Section 3.6.1 that EG could be handled directly.

```
function SAT_EG (φ)
/* determines the set of states satisfying EG φ */
local var X, Y
begin
    Y := SAT(φ);
    X := ∅;
    repeat until X = Y
    begin
        X := Y;
        Y := Y ∩ pre_∃(Y)
    end
    return Y
end
```

Figure 3.37. The pseudo-code for $\mathtt{SAT}_{\mathtt{EG}}$.

3.7.1 Monotone functions

Definition 3.22 Let S be a set of states and $F\colon \mathcal{P}(S) \to \mathcal{P}(S)$ a function on the power set of S.

1. We say that F is monotone iff $X \subseteq Y$ implies $F(X) \subseteq F(Y)$ for all subsets X and Y of S.
2. A subset X of S is called a fixed point of F iff $F(X) = X$.

For an example, let $S \stackrel{\text{def}}{=} \{s_0, s_1\}$ and $F(Y) \stackrel{\text{def}}{=} Y \cup \{s_0\}$ for all subsets Y of S. Since $Y \subseteq Y'$ implies $Y \cup \{s_0\} \subseteq Y' \cup \{s_0\}$, we see that F is monotone. The fixed points of F are all subsets of S containing s_0. Thus, F has two fixed points, the sets $\{s_0\}$ and $\{s_0, s_1\}$. Notice that F has a least ($= \{s_0\}$) and a greatest ($= \{s_0, s_1\}$) fixed point.

An example of a function $G\colon \mathcal{P}(S) \to \mathcal{P}(S)$, which is *not* monotone, is given by

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

So G maps $\{s_0\}$ to $\{s_1\}$ and *all other sets* to $\{s_0\}$. The function G is not monotone since $\{s_0\} \subseteq \{s_0, s_1\}$ but $G(\{s_0\}) = \{s_1\}$ is *not* a subset of $G(\{s_0, s_1\}) = \{s_0\}$. Note that G has *no* fixed points whatsoever.

The reasons for exploring monotone functions on $\mathcal{P}(S)$ in the context of proving the correctness of SAT are:

1. that monotone functions *always* have a least and a greatest fixed point;
2. that the meanings of EG, AF and EU can be expressed via greatest, respectively least, fixed points of monotone functions on $\mathcal{P}(S)$;
3. that these fixed-points can be easily computed, and;
4. that the procedures $\mathtt{SAT}_{\mathtt{EU}}$ and $\mathtt{SAT}_{\mathtt{AF}}$ code up such fixed-point computations, and are correct by item 2.

Notation 3.23 $F^i(X)$ means

$$\underbrace{F(F(\ldots F}_{i \text{ times}}(X)\ldots))$$

Thus, the function F^i is just 'F applied i many times.'

For example, for the function $F(Y) \stackrel{\text{def}}{=} Y \cup \{s_0\}$, we obtain $F^2(Y) = F(F(Y)) = (Y \cup \{s_0\}) \cup \{s_0\} = Y \cup \{s_0\} = F(Y)$. In this case, $F^2 = F$ and therefore $F^i = F$ for all $i \geq 1$. It is not always the case that the sequence of functions $(F^1, F^2, F^3, \ldots)$ stabilises in such a way. For example, this won't happen for the function G defined above (see Exercise 1(d) on page 253). The following fact is a special case of a fundamental insight, often referred to as the Knaster–Tarski Theorem.

Theorem 3.24 Let S be a set $\{s_0, s_1, \ldots, s_n\}$ with $n+1$ elements. If $F\colon \mathcal{P}(S) \to \mathcal{P}(S)$ is a monotone function, then $F^{n+1}(\emptyset)$ is the least fixed point of F and $F^{n+1}(S)$ is the greatest fixed point of F.

PROOF: Since $\emptyset \subseteq F(\emptyset)$, we get $F(\emptyset) \subseteq F(F(\emptyset))$, i.e., $F^1(\emptyset) \subseteq F^2(\emptyset)$, for F is monotone. We can now use mathematical induction to show that

$$F^1(\emptyset) \subseteq F^2(\emptyset) \subseteq F^3(\emptyset) \subseteq \ldots \subseteq F^i(\emptyset)$$

for all $i \geq 1$. In particular, taking $i \stackrel{\text{def}}{=} n+1$, we claim that one of the expressions $F^k(\emptyset)$ above is already a fixed point of F. Otherwise, $F^1(\emptyset)$ needs to contain at least one element (for then $\emptyset \neq F(\emptyset)$). By the same token, $F^2(\emptyset)$ needs to have at least two elements since it must be bigger than $F^1(\emptyset)$. Continuing this argument, we see that $F^{n+2}(\emptyset)$ would have to contain at least $n+2$ many elements. The latter is impossible since S has only $n+1$ elements. Therefore, $F(F^k(\emptyset)) = F^k(\emptyset)$ for some $0 \leq k \leq n+1$, which readily implies that $F^{n+1}(\emptyset)$ is a fixed point of F as well.

Now suppose that X is another fixed point of F. We need to show that $F^{n+1}(\emptyset)$ is a subset of X; but, since $\emptyset \subseteq X$, we conclude $F(\emptyset) \subseteq F(X) = X$, for F is monotone and X a fixed point of F. By induction, we obtain $F^i(\emptyset) \subseteq X$ for all $i \geq 0$. So, for $i \stackrel{\text{def}}{=} n+1$, we get $F^{n+1}(\emptyset) \subseteq X$.

The proof of the statements about the greatest fixed point is dual to the one above. Simply replace $\subseteq$ by $\supseteq$, $\emptyset$ by S and 'bigger' by 'smaller.' $\square$

This theorem about the existence of least and greatest fixed points of monotone functions $F\colon \mathcal{P}(S) \to \mathcal{P}(S)$ not only asserted the existence of such fixed points; it also provided a recipe for computing them, and correctly so. For example, in computing the least fixed point of F, all we have to do is apply F to the empty set $\emptyset$ and keep applying F to the result until the latter becomes invariant under the application of F. The theorem above further ensures that this process is *guaranteed to terminate*. Moreover, we can specify an upper bound $n+1$ to the worst-case number of iterations necessary for reaching this fixed point, assuming that S has $n+1$ elements.

3.7.2 The correctness of $\mathtt{SAT}_{\mathtt{EG}}$

We saw at the end of the last section that $[\![\mathrm{EG}\,\phi]\!] = [\![\phi]\!] \cap \mathrm{pre}_\exists([\![\mathrm{EG}\,\phi]\!])$. This implies that $\mathrm{EG}\,\phi$ is a fixed point of the function $F(X) = [\![\phi]\!] \cap \mathrm{pre}_\exists(X)$. In fact, F is monotone, $\mathrm{EG}\,\phi$ is its greatest fixed point and therefore $\mathrm{EG}\,\phi$ can be computed using Theorem 3.24.

Theorem 3.25 Let F be as defined above and let S have $n+1$ elements. Then F is monotone, $[\![\mathrm{EG}\,\phi]\!]$ is the greatest fixed point of F, and $[\![\mathrm{EG}\,\phi]\!] = F^{n+1}(S)$.

PROOF:

1. In order to show that F is monotone, we take any two subsets X and Y of S such that $X \subseteq Y$ and we need to show that $F(X)$ is a subset of $F(Y)$. Given s_0 such that there is some $s_1 \in X$ with $s_0 \to s_1$, we certainly have $s_0 \to s_1$, where $s_1 \in Y$, for X is a subset of Y. Thus, we showed $\mathrm{pre}_\exists(X) \subseteq \mathrm{pre}_\exists(Y)$ from which we readily conclude that $F(X) = [\![\phi]\!] \cap \mathrm{pre}_\exists(X) \subseteq [\![\phi]\!] \cap \mathrm{pre}_\exists(Y) = F(Y)$.
2. We have already seen that $[\![\mathrm{EG}\,\phi]\!]$ is a fixed point of F. To show that it is the greatest fixed point, it suffices to show here that any set X with $F(X) = X$ has to be contained in $[\![\mathrm{EG}\,\phi]\!]$. So let s_0 be an element of such a fixed point X. We need to show that s_0 is in $[\![\mathrm{EG}\,\phi]\!]$ as well. For that we use the fact that
$$s_0 \in X = F(X) = [\![\phi]\!] \cap \mathrm{pre}_\exists(X)$$
to infer that $s_0 \in [\![\phi]\!]$ and $s_0 \to s_1$ for some $s_1 \in X$; but, since s_1 is in X, we may apply that same argument to $s_1 \in X = F(X) = [\![\phi]\!] \cap \mathrm{pre}_\exists(X)$ and we get $s_1 \in [\![\phi]\!]$ and $s_1 \to s_2$ for some $s_2 \in X$. By mathematical induction, we can therefore construct an infinite path $s_0 \to s_1 \to \cdots \to s_n \to s_{n+1} \to \ldots$ such that $s_i \in [\![\phi]\!]$ for all $i \geq 0$. By the definition of $[\![\mathrm{EG}\,\phi]\!]$, this entails $s_0 \in [\![\mathrm{EG}\,\phi]\!]$.
3. The last item is now immediately accessible from the previous one and Theorem 3.24. □

Now we can see that the procedure $\mathtt{SAT}_{\mathtt{EG}}$ is correctly coded and terminates. First, note that the line $Y := Y \cap \mathrm{pre}_\exists(Y)$ in the procedure $\mathtt{SAT}_{\mathtt{EG}}$ (Figure 3.37) could be changed to $Y := \mathtt{SAT}(\phi) \cap \mathrm{pre}_\exists(Y)$ without changing the effect of the procedure. To see this, note that the first time round the loop, Y *is* $\mathtt{SAT}(\phi)$; and in subsequent loops, $Y \subseteq \mathtt{SAT}(\phi)$, so it doesn't matter whether we intersect with Y or $\mathtt{SAT}(\phi)$[2]. With the change, it is clear that $\mathtt{SAT}_{\mathtt{EG}}$ is calculating the greatest fixed point of F; therefore its correctness follows from Theorem 3.25.

3.7.3 The correctness of $\mathtt{SAT}_{\mathtt{EU}}$

Proving the correctness of $\mathtt{SAT}_{\mathtt{EU}}$ is similar. We start by noting the equivalence $\mathrm{E}[\phi\ \mathrm{U}\ \psi] \equiv \psi \vee (\phi \wedge \mathrm{EX}\,\mathrm{E}[\phi\ \mathrm{U}\ \psi])$ and we write it as $[\![\mathrm{E}[\phi\ \mathrm{U}\ \psi]]\!] = [\![\psi]\!] \cup ([\![\phi]\!] \cap \mathrm{pre}_\exists[\![\mathrm{E}[\phi\ \mathrm{U}\ \psi]]\!])$. That tells us that $[\![\mathrm{E}[\phi\ \mathrm{U}\ \psi]]\!]$ is a fixed point of the function $G(X) = [\![\psi]\!] \cup ([\![\phi]\!] \cap \mathrm{pre}_\exists(X))$. As before, we can prove that this function is monotone. It turns out that $[\![\mathrm{E}[\phi\ \mathrm{U}\ \psi]]\!]$ is its *least* fixed point and that the function $\mathtt{SAT}_{\mathtt{EU}}$ is actually computing it in the manner of Theorem 3.24.

Theorem 3.26 Let G be defined as above and let S have $n+1$ elements. Then G is monotone, $[\![\mathrm{E}(\phi\ \mathrm{U}\ \psi)]\!]$ is the least fixed point of G, and we have $[\![\mathrm{E}(\phi\ \mathrm{U}\ \psi)]\!] = G^{n+1}(\emptyset)$.

[2] If you are sceptical, try computing the values $Y_0, Y_1, Y_2, \ldots$, where Y_i represents the value of Y after i iterations round the loop. The program before the change computes as follows:

$$\begin{aligned} Y_0 &= \mathtt{SAT}(\phi) \\ Y_1 &= Y_0 \cap \mathrm{pre}_\exists(Y_0) \\ Y_2 &= Y_1 \cap \mathrm{pre}_\exists(Y_1) \\ &= Y_0 \cap \mathrm{pre}_\exists(Y_0) \cap \mathrm{pre}_\exists(Y_0 \cap \mathrm{pre}_\exists(Y_0)) \\ &= Y_0 \cap \mathrm{pre}_\exists(Y_0 \cap \mathrm{pre}_\exists(Y_0)). \end{aligned}$$

The last of these equalities follows from the monotonicity of $\mathrm{pre}_\exists$.

$$\begin{aligned} Y_3 &= Y_2 \cap \mathrm{pre}_\exists(Y_2) \\ &= Y_0 \cap \mathrm{pre}_\exists(Y_0 \cap \mathrm{pre}_\exists(Y_0)) \cap \mathrm{pre}_\exists(Y_0 \cap \mathrm{pre}_\exists(Y_0 \cap \mathrm{pre}_\exists(Y_0))) \\ &= Y_0 \cap \mathrm{pre}_\exists(Y_0 \cap \mathrm{pre}_\exists(Y_0 \cap \mathrm{pre}_\exists(Y_0))). \end{aligned}$$

Again the last one follows by monotonicity. Now look at what the program does after the change:

$$\begin{aligned} Y_0 &= \mathtt{SAT}(\phi) \\ Y_1 &= \mathtt{SAT}(\phi) \cap \mathrm{pre}_\exists(Y_0) \\ &= Y_0 \cap \mathrm{pre}_\exists(Y_0) \\ Y_2 &= Y_0 \cap \mathrm{pre}_\exists(Y_1) \\ Y_3 &= Y_0 \cap \mathrm{pre}_\exists(Y_1) \\ &= Y_0 \cap \mathrm{pre}_\exists(Y_0 \cap \mathrm{pre}_\exists(Y_0)). \end{aligned}$$

A formal proof would follow by induction on i.