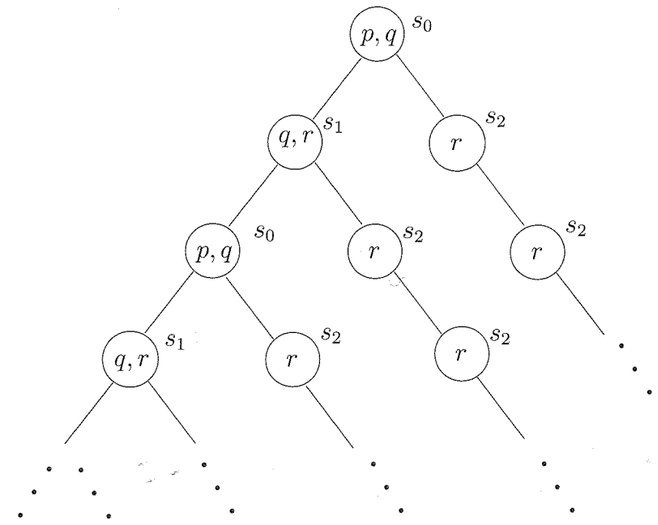

Figure 3.5. Unwinding the system of Figure 3.3 as an infinite tree of all computation paths beginning in a particular state.

It is useful to visualise all possible computation paths from a given state s by unwinding the transition system to obtain an infinite computation tree. For example, if we unwind the state graph of Figure 3.3 for the designated starting state s_0, then we get the infinite tree in Figure 3.5. The execution paths of a model $\mathcal{M}$ are explicitly represented in the tree obtained by unwinding the model.

Definition 3.6 Let $\mathcal{M} = (S, \rightarrow, L)$ be a model and $\pi = s_1 \rightarrow \ldots$ be a path in $\mathcal{M}$. Whether π satisfies an LTL formula is defined by the satisfaction relation $\models$ as follows:

1. $\pi \models \top$
2. $\pi \not\models \bot$
3. $\pi \models p$ iff $p \in L(s_1)$
4. $\pi \models \neg\phi$ iff $\pi \not\models \phi$
5. $\pi \models \phi_1 \wedge \phi_2$ iff $\pi \models \phi_1$ and $\pi \models \phi_2$
6. $\pi \models \phi_1 \vee \phi_2$ iff $\pi \models \phi_1$ or $\pi \models \phi_2$
7. $\pi \models \phi_1 \rightarrow \phi_2$ iff $\pi \models \phi_2$ whenever $\pi \models \phi_1$
8. $\pi \models \mathrm{X}\,\phi$ iff $\pi^2 \models \phi$
9. $\pi \models \mathrm{G}\,\phi$ iff, for all $i \geq 1$, $\pi^i \models \phi$

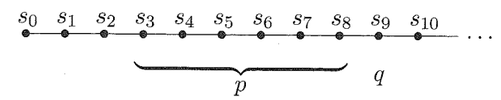

Figure 3.6. An illustration of the meaning of Until in the semantics of LTL. Suppose p is satisfied at (and only at) s_3, s_4, s_5, s_6, s_7, s_8 and q is satisfied at (and only at) s_9. Only the states s_3 to s_9 each satisfy $p\,\mathrm{U}\,q$ along the path shown.

10. $\pi \models \mathrm{F}\,\phi$ iff there is some $i \geq 1$ such that $\pi^i \models \phi$
11. $\pi \models \phi\,\mathrm{U}\,\psi$ iff there is some $i \geq 1$ such that $\pi^i \models \psi$ and for all $j = 1, \ldots, i-1$ we have $\pi^j \models \phi$
12. $\pi \models \phi\,\mathrm{W}\,\psi$ iff either there is some $i \geq 1$ such that $\pi^i \models \psi$ and for all $j = 1, \ldots, i-1$ we have $\pi^j \models \phi$; or for all $k \geq 1$ we have $\pi^k \models \phi$
13. $\pi \models \phi\,\mathrm{R}\,\psi$ iff either there is some $i \geq 1$ such that $\pi^i \models \phi$ and for all $j = 1, \ldots, i$ we have $\pi^j \models \psi$, or for all $k \geq 1$ we have $\pi^k \models \psi$.

Clauses 1 and 2 reflect the facts that $\top$ is always true, and $\bot$ is always false. Clauses 3–7 are similar to the corresponding clauses we saw in propositional logic. Clause 8 removes the first state from the path, in order to create a path starting at the 'next' (second) state.

Notice that clause 3 means that atoms are evaluated in the first state along the path in consideration. However, that doesn't mean that all the atoms occuring in an LTL formula refer to the first state of the path; if they are in the scope of a temporal connective, e.g., in $\mathrm{G}\,(p \rightarrow \mathrm{X}\,q)$, then the calculation of satisfaction involves taking suffices of the path in consideration, and the atoms refer to the first state of those suffices.

Let's now look at clauses 11–13, which deal with the binary temporal connectives. U, which stands for 'Until,' is the most commonly encountered one of these. The formula $\phi_1\,\mathrm{U}\,\phi_2$ holds on a path if it is the case that ϕ_1 holds continuously *until* ϕ_2 holds. Moreover, $\phi_1\,\mathrm{U}\,\phi_2$ actually demands that ϕ_2 *does* hold in some future state. See Figure 3.6 for illustration: each of the states s_3 to s_9 satisfies $p\,\mathrm{U}\,q$ along the path shown, but s_0 to s_2 don't.

The other binary connectives are W, standing for 'Weak-until,' and R, standing for 'Release.' Weak-until is just like U, except that $\phi\,\mathrm{W}\,\psi$ does not require that ψ is eventually satisfied along the path in question, which is required by $\phi\,\mathrm{U}\,\psi$. Release R is the dual of U; that is, $\phi\,\mathrm{R}\,\psi$ is equivalent to $\neg(\neg\phi\,\mathrm{U}\,\neg\psi)$. It is called 'Release' because clause 11 determines that ψ must remain true up to and including the moment when ϕ becomes true (if there is one); ϕ 'releases' ψ. R and W are actually quite similar; the differences are that they swap the roles of ϕ and ψ, and the clause for W has an $i-1$

where R has i. Since they are similar, why do we need both? We don't; they are interdefinable, as we will see later. However, it's useful to have both. R is useful because it is the dual of U, while W is useful because it is a weak form of U.

Note that neither the strong version (U) or the weak version (W) of until says anything about what happens after the until has been realised. This is in contrast with some of the readings of 'until' in natural language. For example, in the sentence 'I smoked until I was 22' it is not only expressed that the person referred to continually smoked up until he or she was 22 years old, but we also would interpret such a sentence as saying that this person gave up smoking from that point onwards. This is different from the semantics of until in temporal logic. We could express the sentence about smoking by combining U with other connectives; for example, by asserting that it was once true that $s \,\mathrm{U}\, (t \wedge \mathrm{G}\, \neg s)$, where s represents 'I smoke' and t represents 'I am 22.'

Remark 3.7 Notice that, in clauses 9–13 above, the future includes the present. This means that, when we say 'in all future states,' we are including the present state as a future state. It is a matter of convention whether we do this, or not. As an exercise, you may consider developing a version of LTL in which the future excludes the present. A consequence of adopting the convention that the future shall include the present is that the formulas $\mathrm{G}\, p \to p$, $p \to q \,\mathrm{U}\, p$ and $p \to \mathrm{F}\, p$ are true in every state of every model.

So far we have defined a satisfaction relation between paths and LTL formulas. However, to verify systems, we would like to say that a model as a whole satisfies an LTL formula. This is defined to hold whenever *every* possible execution path of the model satisfies the formula.

Definition 3.8 Suppose $\mathcal{M} = (S, \to, L)$ is a model, $s \in S$, and ϕ an LTL formula. We write $\mathcal{M}, s \vDash \phi$ if, for every execution path π of $\mathcal{M}$ starting at s, we have $\pi \vDash \phi$.

If $\mathcal{M}$ is clear from the context, we may abbreviate $\mathcal{M}, s \vDash \phi$ by $s \vDash \phi$. It should be clear that we have outlined the formal foundations of a procedure that, given ϕ, $\mathcal{M}$ and s, can check whether $\mathcal{M}, s \vDash \phi$ holds. Later in this chapter, we will examine algorithms which implement this calculation. Let us now look at some example checks for the system in Figures 3.3 and 3.5.

1. $\mathcal{M}, s_0 \vDash p \wedge q$ holds since the atomic symbols p and q are contained in the node of s_0: $\pi \vDash p \wedge q$ for *every* path π beginning in s_0.

2. $\mathcal{M}, s_0 \vDash \neg r$ holds since the atomic symbol r is *not* contained in node s_0.
3. $\mathcal{M}, s_0 \vDash \top$ holds by definition.
4. $\mathcal{M}, s_0 \vDash \mathrm{X}\, r$ holds since all paths from s_0 have either s_1 or s_2 as their next state, and each of those states satisfies r.
5. $\mathcal{M}, s_0 \vDash \mathrm{X}\, (q \wedge r)$ does not hold since we have the rightmost computation path $s_0 \to s_2 \to s_2 \to s_2 \to \ldots$ in Figure 3.5, whose second node s_2 contains r, but not q.
6. $\mathcal{M}, s_0 \vDash \mathrm{G}\, \neg(p \wedge r)$ holds since all computation paths beginning in s_0 satisfy $\mathrm{G}\, \neg(p \wedge r)$, i.e. they satisfy $\neg(p \wedge r)$ in each state along the path. Notice that $\mathrm{G}\, \phi$ holds in a state if, and only if, ϕ holds in all states reachable from the given state.
7. For similar reasons, $\mathcal{M}, s_2 \vDash \mathrm{G}\, r$ holds (note the s_2 instead of s_0).
8. For any state s of $\mathcal{M}$, we have $\mathcal{M}, s \vDash \mathrm{F}\, (\neg q \wedge r) \to \mathrm{F}\, \mathrm{G}\, r$. This says that if any path π beginning in s gets to a state satisfying $(\neg q \wedge r)$, then the path π satisfies $\mathrm{F}\, \mathrm{G}\, r$. Indeed this is true, since if the path has a state satisfying $(\neg q \wedge r)$ then (since that state must be s_2) the path does satisfy $\mathrm{F}\, \mathrm{G}\, r$. Notice what $\mathrm{F}\, \mathrm{G}\, r$ says about a path: eventually, you have continuously r.
9. The formula $\mathrm{G}\, \mathrm{F}\, p$ expresses that p occurs along the path in question infinitely often. Intuitively, it's saying: no matter how far along the path you go (that's the G part) you will find you still have a p in front of you (that's the F part). For example, the path $s_0 \to s_1 \to s_0 \to s_1 \to \ldots$ satisfies $\mathrm{G}\, \mathrm{F}\, p$. But the path $s_0 \to s_2 \to s_2 \to s_2 \to \ldots$ doesn't.
10. In our model, if a path from s_0 has infinitely many ps on it then it must be the path $s_0 \to s_1 \to s_0 \to s_1 \to \ldots$, and in that case it also has infinitely many rs on it. So, $\mathcal{M}, s_0 \vDash \mathrm{G}\, \mathrm{F}\, p \to \mathrm{G}\, \mathrm{F}\, r$. But it is not the case the other way around! It is not the case that $\mathcal{M}, s_0 \vDash \mathrm{G}\, \mathrm{F}\, r \to \mathrm{G}\, \mathrm{F}\, p$, because we can find a path from s_0 which has infinitely many rs but only one p.

3.2.3 Practical patterns of specifications

What kind of practically relevant properties can we check with formulas of LTL? We list a few of the common patterns. Suppose atomic descriptions include some words such as busy and requested. We may require some of the following properties of real systems:

- It is impossible to get to a state where started holds, but ready does not hold:
  $G\neg(\mathsf{started} \wedge \neg\mathsf{ready})$
  The negation of this formula expresses that it *is possible* to get to such a state, but this is only so if interpreted on paths ($\pi \vDash \phi$). We cannot assert such a possibility if interpreted on states ($s \vDash \phi$) since we cannot express the existence of paths; for that interpretation, the negation of the formula above asserts that *all* paths will eventually get to such a state.

- For any state, if a request (of some resource) occurs, then it will eventually be acknowledged:
  G (requested → F acknowledged).
- A certain process is enabled infinitely often on every computation path:
  G F enabled.
- Whatever happens, a certain process will eventually be permanently deadlocked:
  F G deadlock.
- If the process is enabled infinitely often, then it runs infinitely often.
  G F enabled → G F running.
- An upwards travelling lift at the second floor does not change its direction when it has passengers wishing to go to the fifth floor:
  G (floor2 ∧ directionup ∧ ButtonPressed5 → (directionup U floor5))
  Here, our atomic descriptions are boolean expressions built from system variables, e.g., floor2.

There are some things which are *not* possible to say in LTL, however. One big class of such things are statements which assert the existence of a path, such as these ones:

- From any state *it is possible* to get to a restart state (i.e., there is a path from all states to a state satisfying restart).
- The lift *can* remain idle on the third floor with its doors closed (i.e., from the state in which it is on the third floor, there is a path along which it stays there).

LTL can't express these because it cannot directly assert the existence of paths. In Section 3.4, we look at Computation Tree Logic (CTL) which has operators for quantifying over paths, and can express these properties.

3.2.4 Important equivalences between LTL formulas

Definition 3.9 We say that two LTL formulas ϕ and ψ are semantically equivalent, or simply equivalent, writing $\phi \equiv \psi$, if for all models $\mathcal{M}$ and all paths π in $\mathcal{M}$: $\pi \models \phi$ iff $\pi \models \psi$.

The equivalence of ϕ and ψ means that ϕ and ψ are semantically interchangeable. If ϕ is a subformula of some bigger formula χ, and $\psi \equiv \phi$, then we can make the substitution of ψ for ϕ in χ without changing the meaning of χ. In propositional logic, we saw that $\wedge$ and $\vee$ are duals of each other, meaning that if you push a $\neg$ past a $\wedge$, it becomes a $\vee$, and vice versa:

$$\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi \qquad \neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi.$$

(Because $\wedge$ and $\vee$ are binary, pushing a negation downwards in the parse tree past one of them also has the effect of duplicating that negation.)

Similarly, F and G are duals of each other, and X is dual with itself:

$$\neg \mathrm{G}\,\phi \equiv \mathrm{F}\,\neg\phi \qquad \neg \mathrm{F}\,\phi \equiv \mathrm{G}\,\neg\phi \qquad \neg \mathrm{X}\,\phi \equiv \mathrm{X}\,\neg\phi.$$

Also U and R are duals of each other:

$$\neg(\phi \,\mathrm{U}\, \psi) \equiv \neg\phi \,\mathrm{R}\, \neg\psi \qquad \neg(\phi \,\mathrm{R}\, \psi) \equiv \neg\phi \,\mathrm{U}\, \neg\psi.$$

We should give formal proofs of these equivalences. But they are easy, so we leave them as an exercise to the reader. 'Morally' there ought to be a dual for W, and you can invent one if you like. Work out what it might mean, and then pick a symbol based on the first letter of the meaning. However, it might not be very useful.

It's also the case that F distributes over $\vee$ and G over $\wedge$, i.e.,

$$\mathrm{F}\,(\phi \vee \psi) \equiv \mathrm{F}\,\phi \vee \mathrm{F}\,\psi$$
$$\mathrm{G}\,(\phi \wedge \psi) \equiv \mathrm{G}\,\phi \wedge \mathrm{G}\,\psi.$$

Compare this with the quantifier equivalences in Section 2.3.2. But F does *not* distribute over $\wedge$. What this means is that there is a model with a path which distinguishes $\mathrm{F}\,(\phi \wedge \psi)$ and $\mathrm{F}\,\phi \wedge \mathrm{F}\,\psi$, for some ϕ, ψ. Take the path $s_0 \to s_1 \to s_0 \to s_1 \to \ldots$ from the system of Figure 3.3, for example; it satisfies $\mathrm{F}\,p \wedge \mathrm{F}\,r$ but it doesn't satisfy $\mathrm{F}\,(p \wedge r)$.

Here are two more equivalences in LTL:

$$\mathrm{F}\,\phi \equiv \top \,\mathrm{U}\, \phi \qquad \mathrm{G}\,\phi \equiv \bot \,\mathrm{R}\, \phi.$$

The first one exploits the fact that the clause for Until states two things: the second formula ϕ must become true; and until then, the first formula $\top$ must hold. So, if we put 'no constraint' for the first formula, it boils down to asking that the second formula holds, which is what F asks. (The formula $\top$ represent 'no constraint.' If you ask me to bring it about that $\top$ holds, I need do nothing, it enforces no constraint. In the same sense, $\bot$ is 'every constraint.' If you ask me to bring it about that $\bot$ holds, I'll have to meet every constraint there is, which is impossible.)

The second formula, that $\mathrm{G}\,\phi \equiv \bot \,\mathrm{R}\, \phi$, can be obtained from the first by putting a $\neg$ in front of each side, and applying the duality rules. Another more intuitive way of seeing this is to recall the meaning of 'release:' $\bot$ releases ϕ, but $\bot$ will never be true, so ϕ doesn't get released.

Another pair of equivalences relates the strong and weak versions of Until, U and W. Strong until may be seen as weak until plus the constraint that the eventuality must actually occur:

$$\phi \,\mathrm{U}\, \psi \equiv \phi \,\mathrm{W}\, \psi \wedge \mathrm{F}\,\psi\,. \tag{3.2}$$