Finally, a note about how Prolog matching sometimes differs from the unification used in Resolution. Most Prolog systems will allow you to satisfy goals like:

equal(X, X).
?- equal(foo(Y), Y).

that is, they will allow you to match a term against an uninstantiated subterm of itself. In this example, foo(Y) is matched against Y, which appears within it. As a result, Y will stand for foo(Y), which is foo(foo(Y)) (because of what Y stands for), which is foo(foo(foo(Y))), and so on. So Y ends up standing for some kind of infinite structure.

Note that, whereas they may allow you to construct something like this, most Prolog systems will not be able to write it out at the end. According to the formal definition of Unification, this kind of “infinite term” should never come to exist. Thus Prolog systems that allow a term to match an uninstantiated subterm of itself do not act correctly as Resolution theorem provers. In order to make them do so, we would have to add a check that a variable cannot be instantiated to something containing itself. Such a check, an occurs check, would be straightforward to implement, but would slow down the execution of Prolog programs considerably. Since it would only affect very few programs, most implementors have simply left it out 1.