Prover9:

Basic information:
Prover9, the successor of Otter, can operate autonomously however it is mainly controlled with manual user interaction. It accepts statements in first-order logic, equational logic or with clauses (a disjunction of literals such as $l_1 \lor l_2 \ldots \lor l_n$). Rather than directly translating everything to clauses, like its predecessor, Prover9 removes the key differences between “clauses” and “formulas”. Instead, “clauses” become a subset of “formulas” – this means that the formulas can have their own free variables. Prover9 also supports a formula goal, which becomes negated in attempt for a proof. Prover9 will try and automatically find a proof by default (rather than having to have it explicitly set as its predecessor).

If Prover9 is simply given a set of formulas, clauses or both it will look at the clauses and decide which inference rules and processing operations to use. However this automatic choice is optional; it is possible to specify your own rules; search limits, clause weight values etc. An example of this is:

```
clear(auto) -> clear(auto_limits).
clear(auto_limits) -> assign(max_weight, 2147483647).
clear(auto_limits) -> assign(sos_limit, -1).
```

This manual example means that Prover9 will keep searching, with no limit, for new clauses and discards inferred clauses according to the maximum weight parameter (never applied to initial clauses).

Mace4 works together with Prover9 and searches for finite structures of first-order and equational statements in order to attempt to find a counterexample. It can be very useful running Mace4 on a conjecture before running Prover9; however they can run simultaneously on the same input.

Prover9 will always prove by contradiction. No goals need to be specified (it will search for a contradiction among the assumptions). Also no assumptions need to be specified if the entire statement of the assumption is in the goal.

Quick rundown of using Prover9:
The syntax is fairly straight forward. The only special observations worth of noting are:

- Formulas should end with a period “.”
- Variable should start with a lowercase letter between ‘u’ and ‘z’
- Logic connectives are - | & -> <-> all exists.
- Parentheses can be used as expected, the output will only show the minimum number required.

Examples:
This shows a GUI implementation of Prover9 setup with the example: $\forall x. ( P(x) \& Q(x) ) \vdash \forall x. P(x) \& \forall x. Q(x)$
As you can see, using Mace4 to try and find a counterexample results in an exhausted search (no counterexample found).

The output of Mace4 can, however, be used for debugging the proof & often provides much useful information.

<Right> Here is the output of Prover9. As you can see it took 0.01 seconds and the proof was found completely automatically with 7 lines and 3 levels.

Lines 1 contain the assumption. Line 2 contains the goal. This is resolved by line 7. All output can be saved as expected.

This example by no means harnesses the full power and flexibility of Prover9.

<Above> The following shows an interesting example of how Prover9 can be used with a slightly larger example. In this case, the proof is calculated in 0.03 seconds with 74 lines and 10 levels.
Analysis:
Prover9 with Mace4 is an extremely flexible and powerful theorem prover. It can handle cases of equality and non-equality, it can search for counter examples, it can handle clauses, first-order-logic, equational logic. It can search for multiple goals (and attempt to prove all of them).

In its disadvantage, proofs found by Prover9 are often hard to understand and longer than required. Research is currently underway to improve this; specifically looking at methods of proof transformation.

Prover9 can be confusing in the standard command line format, especially for new users confused about Mace4 and how to make it manually interactive. However, the GUI distribution is extremely easy to use and comes in precompiled Windows, Macintosh and Linux binaries. It even has pre-bundled sample examples ready to be run. These make the learning curve very fast & installation is easy.

ProofWeb (Coq, Isabelle):

Basic information:
ProofWeb is a web interface with similar design to Coqide. It supports a complete graphical frontend for Coq and Isabelle/HOL assisted theorem provers. It is designed for Mozilla/Gecko based web browsers and doesn’t require installation. ProofWeb effectively has many features, but the individual learning curve for each prover can be very steep & lengthy. It is not recommended for new users.

An anonymous login can be found here: http://prover.cs.ru.nl/login.php [See References]

→ Usage information:
Special observations include:

- Using ProofWeb with Coq requires the user to first state their theorem followed by a period “.”
- It should then be followed by the term “Proof.” and finish with “Qed.”
- Between, the user tries to manually resolve the proof while the output is showed at each stage.
- The user can backtrack using the “up” and “down” arrows and follow the output steps on the upper-right screen.
- If mistakes, syntax errors or impossible tactics are made, then the output will explain what is incorrect as well as giving a useful indication as to where the mistake is made.
- Subgoals are shown while the user tries to reach them.
- Syntax varies between theorem prover and this has to be learnt manually. There are many different ways to prove a particular theorem such as forwards, backwards and mixed reasoning tactics. This is entirely up to the user at all times.
- Parentheses can be used as expected, the output will only show the minimum number required.

Examples:
The following screenshot shows the ProofWeb interface running Coq. The green up and down arrows are used to navigate what is currently being processed. Previously processed information is highlighted in green and this cannot be edited. The remaining text is designed to be edited by the user as they prove it while the upper right display shows how the proof is coming along.
The screenshot above shows an example of a completed proof, and an incomplete one. As the user steps to the proof, the output displays:

1 subgoal

all x, all y, (R(y, y) / \ x = y -> R(y, x))

In this example, the user then decides to apply the all_i rule on x giving the output:

1 subgoal

x : D

all y, (R(y, y) / \ x = y -> R(y, x))

The user then continues the assisted proof and reaches a maximum level with three subgoals after the con_e2 (R(y,y)). command as shown:

3 subgoals

x : D
y : D
H : R(y, y) / \ x = y

R(y, y) / \ x = y

subgoal 2 is:
x = x
subgoal 3 is:
R(y, y)

When completed, the output displays “Proof Completed”.

Analysis:

ProofWeb consists of multiple tools housed in one graphical interface. Because of this, using ProofWeb and running examples has a learning curve of around 1 minute. However using the theorem provers within ProofWeb is much more complicated; requires much more training and each has a learning curve of approximately 1 month.
Its strengths are with its portability & functionality as a teaching tool; students can create accounts and work on proofs individually, while lecturers can manage multiple students work and grade each student with ease. Both Coq and Isabelle can be difficult and impractical to install; for example, Coq requires a Linux operating system or an environmental windows shell/path extension manager such as Moscow ML; whereas ProofWeb requires none of this and works directly from a browser. Coq provides a powerful interface for approaching difficult theorems and proofs. An example of where it has been outside of teaching is with creating a surveying proof for the four color theorem (where no two colours are the same on adjacent regions) in any map. ProofWeb has the potential power of Coq and Isabelle combined.

The disadvantages of ProofWeb mainly lie within Coq and Isabelle themselves. Both theorem provers have difficult learning curves and require a large amount of expertise to operate and use effectively. The learning curve is around 1 month for each tool, and requires a fair amount of background reading. Also, being assisted interactive tools, a proof may require much time, many attempts, backtracking and starting over – it is common that the “best” or most elegant solution is not the first one found.

Summary and comparison:
To summarize and compare the tools against each other, each have their own individual advantages and disadvantages as mentioned previously in their analysis sections. Prover9 provides a fast and flexible way to achieve quick and potentially powerful results, whereas Coq also provides a flexible way and has already shown to solve some famous real world proofs. Prover9 will be useful for me in the future, along with Mace4, to aid research, ideas, or theories due to its speed and ability to search for counter examples.

I do not see ProofWeb, with either Coq or Isabelle, as a practical theorem prover. It simply takes too much time and effort in becoming familiar with the syntax in order to use it effectively. It is true that Coq has been used with beneficial outcome in the past; to prove some very important problems, but with the newer generation of automated proof tools under heavy development, I think that theorem provers such as Prover9 are much more practical.

However, I do think that ProofWeb would be a better teaching aid for students, as it forces them to have an understanding of predicates, functions and general logic.

References:

