Simplified Cheat Sheet Predicate Logic

Martin Henz, 15/3/2013

Bottom-up

General tactic on bottom-up reasoning: When you are proving a goal, and you think you need and can prove a hypothesis ϕ, you can add it via `assert` ϕ. Coq will then ask you to first prove ϕ. After that, you can work on your original goal, now with the additional hypothesis of ϕ. This is a special case of $\rightarrow e$, see below.

$\top i$: `trivial.`

$\wedge i$: `split.`

$\wedge e_1$: Let us call the goal g. Here is how to use $\wedge e_1$ bottom-up, in case you would ever need it: `assert` g `/\` ϕ. prove conjunction, `destruct H1. apply H2.` where `H1` is the conjunction and `H2` is g.

$\wedge e_2$: Let us call the goal g. Here is how to use $\wedge e_2$ bottom-up, in case you would ever need it: `assert` ϕ `/\` g. prove conjunction, `destruct H1. apply H2.` where `H1` is the conjunction and `H2` is g.

$\vee i_1$: `left.`

$\vee i_2$: `right.`

$\vee e$: `destruct H.`

$\rightarrow i$: `intro.`

$\rightarrow e$: `apply H.` (`H` is the implication)

$\rightarrow e$: A variant of the rule allows you to prove a goal ψ, by proving first ϕ, and then $\phi \rightarrow \psi$: `assert` ϕ `.`, then prove ϕ, and finally prove goal ψ using ϕ

$\neg e$: `assert` ϕ `/\ ~` ϕ. `split.` prove ϕ and $\neg\phi$ separately, then use `destruct H1. contradiction H2.`, where `H1` is the asserted conjunction, and `H2` is one part of it.

$\neg i$: `unfold not. intro.`

$\bot e$: `exfalso.`

$\neg\neg e$: Let us call the goal g. Here is how to use $\neg\neg e$ bottom-up, in case you would ever need it: `assert (~ ~` g`).` prove $\neg\neg g$. Now use: `tauto.` equality `H` from right to left)

Derived rule: LEM $+ \vee e$: `LEM (` ϕ `).`

Top-down

Coq allows you to apply some rules within the hypotheses, which makes many proofs a lot shorter. Here are some common uses of top-down reasoning:

$\rightarrow e$: `spec H1 H2.` (`H1` is the implication)

$\neg i$: `unfold not in H.`

$\wedge i$: `destruct H.`