lemma

The *lemma* command makes available the predicate of a named conjecture from the specification as a new antecedent. That conjecture becomes a new sub-goal in the proof.

$$\frac{\vdash ?\ p \qquad \mid p \vdash ?}{\vdash ?}$$

For the *lemma* command to be applicable, the named conjecture must conform to the syntax of a draft Standard Z conjecture, i.e. have only one consequent predicate after the turnstile and nothing but any generic parameters before the turnstile. The conjecture must have a name, as otherwise it would not be possible to record a script of the proof step.

Given a suitable named conjecture, the *lemma* command is applicable when a whole goal is selected in the left window or crossed in the specification window whilst the whole conjecture is selected and inspected in the specification window. Instantiations for any generic parameters of the conjecture are prompted for in dialogue boxes. As the lemma is introduced as a new antecedent predicate, these instantiation expressions may refer to the goal's generic parameters and declarations in the hypothesis, but not to any more-local declarations. To ease correction of keyboard mistakes, each dialogue box is primed with any previous response given for a similarly named generic parameter in a *lemma* step.

As an alternative to typing instantiations into dialogue boxes, the *lemma* com-

mand is also applicable when an expression in a goal is selected and an expression in a named conjecture is selected and inspected, so long as unification of the types of those two expressions suffices to determine an instantiation of the conjecture's generic parameters. In this case, take care to select an expression in the goal that will produce the desired instantiation of the conjecture; often the conjecture will concern a free variable, and selecting the name of that variable in the goal and in the conjecture will be appropriate.

The laws presented in the toolkit are suitable conjectures for use with the *lemma* command. The quickest way to prove such laws should be to use the *apply tactic* command to replay a recorded script of how it was proved before.

See also the *cut conjoined* and *cut disjoined* and *cut apart* and *rewrite by rule* commands.

1. Tactic examples

"lemma" "lemma_name" e_1 e_2

This example applies the *lemma* command using the lemma named "*lemma_name*" and determines the instantiations of any generic parameters by unification of the types of expressions e_1 in the lemma and e_2 in the goal.

"lemma" "lemma_name" 0 0

This example applies the *lemma* command using the lemma named "*lemma_name*", but works only if the lemma is not generic.

"lemma" "lemma_name" "inst" 0 0

This example applies the *lemma* command using the lemma named "*lemma_name*", with the string "*inst*" giving an expression that provides the instantiation of the parameters of the generic lemma; if the lemma has several generic parameters, this expression must be a tuple extension.

IT 17-Apr-2000