rewrite by antecedent

The *rewrite by antecedent* command applies a rewrite rule to an expression or a predicate in a goal.

The expression or predicate is selected first, and the rewrite rule second.

In the expression case, the rewrite rule must be a predicate in one of the following forms.

$$pattern = replacement$$
$$\forall\, t \bullet pattern = replacement$$

In the predicate case, the rewrite rule must be a predicate in one of the following forms.

$$pattern \Leftrightarrow replacement$$
$$pattern \qquad (\Leftrightarrow true\ replacement)$$
$$\neg\ pattern \qquad (\Leftrightarrow false\ replacement)$$
$$\forall\, t \bullet pattern \Leftrightarrow replacement$$
$$\forall\, t \bullet pattern \qquad (\Leftrightarrow true\ replacement)$$
$$\forall\, t \bullet \neg\ pattern \qquad (\Leftrightarrow false\ replacement)$$

The rewrite rule must be an antecedent of the expression or predicate, meaning that it must be a conjunct of a surrounding conjunction predicate, or of the left operand of a surrounding implication predicate, or of the | part of a surrounding

schema text, or of the $\bullet$ part of a surrounding (unique) existential quantification predicate, or of an antecedent predicate of the surrounding goal. The following is an attempt to formalize this, in which r is the rewrite rule appearing as a conjunct of a larger predicate $r \land q$ (not necessarily the first conjunct), and f is the selected formula appearing within a larger predicate $p(f)$.

$$
\begin{array}{l}
r \land p(f) \\
r \land q \Rightarrow p(f) \\
\ldots \mid r \land q \bullet p(f) \\
\exists\, ds \mid p(f) \bullet r \land q \\
\exists_1\, ds \mid p(f) \bullet r \land q
\end{array}
$$

The *pattern* part of the rewrite rule must match the selected expression or predicate, interpreting the variables declared by any schema text t as jokers for this match.

In the case of a quantified rewrite rule, side-conditions arise from any declarations and $\mid$ part of the schema text t. These side-conditions must be automatically decidable for the command to be applicable. Currently, the *simplification tac* command is used as this decision procedure.

The expression or predicate is replaced by the corresponding instantiation of the *replacement*.

Unlike *rewrite by rule* and *rewrite by section*, the rewrite rule must match the entire expression or predicate, and only one rewrite is done.

This command differs from the *Leibniz* command in the following ways. Rewriting

is always done left-to-right, never right-to-left (see *commutation*). The equality or equivalence used as the rewrite rule can be within a quantified predicate. Only one formula can be rewritten by a use of this command (the side-conditions for multiple rewrites would have been too awkward to manage, but this might change in future). The selections must be within a goal, not any other form of paragraph (because there is always somewhere within a goal to put the side-conditions, again this might change soon).

A trace of the instantiated rewrites can be made to appear in the shell window by invoking cadiz with the option -dw.

1. Tactic examples

"rewrite by antecedent" e p "rewrite by antecedent" p_1 p_2

The first example rewrites expression e using predicate p. The second example rewrites predicate p_1 using predicate p_2.

IT 30-Nov-2000