# absorption

/Reference manual/Z-related commands/In situ replacement commands

The *absorption* command performs simplifications that typically remove parts of formulae or at least make them smaller. There are two categories of absorptions: elementary ones that are built-in to cadiz, and ones that are coded explicitly as Z rewrite rules. An example of the latter is as follows.

union_absorption ==

$$[X] \vdash ?\ \forall S : \mathbb{P}\, X \bullet S \cup \ = S$$

The form of a rewrite rule must be as specified in *rewrite by rule*, and moreover its name must have *absorption* as a sub-string. The effect of a rewrite rule is also explained in *rewrite by rule*.

The built-in elementary absorptions are as follows. They take precedence over any matching explicit rewrite rule. In each of the following subsections, the earliest applicable elementary simplification in the list is the one that is applied.

## 1. Contents of this page

# 2. Predicates

## 2.1. Negations

$$\begin{array}{rcl} \neg\, \mathit{false} & \implies & \mathit{true} \\ \neg\, \mathit{true} & \implies & \mathit{false} \\ \neg\, \neg\, p & \implies & p \end{array}$$

## 2.2. Conjunctions

$$
\begin{array}{rcl}
p \land \mathit{false} & \implies & \mathit{false} \\
\mathit{false} \land p & \implies & \mathit{false} \\
p \land \neg\, p & \implies & \mathit{false} \\
\neg\, p \land p & \implies & \mathit{false} \\
p \land \mathit{true} & \implies & p \\
\mathit{true} \land p & \implies & p \\
p \land p & \implies & p
\end{array}
$$

$$
\begin{array}{rcl}
p \ \mathit{NL}\ \mathit{false} & \implies & \mathit{false} \\
\mathit{false}\ \mathit{NL}\ p & \implies & \mathit{false} \\
p \ \mathit{NL}\ \neg\, p & \implies & \mathit{false} \\
\neg\, p\ \mathit{NL}\ p & \implies & \mathit{false} \\
p \ \mathit{NL}\ \mathit{true} & \implies & p \\
\mathit{true}\ \mathit{NL}\ p & \implies & p \\
p \ \mathit{NL}\ p & \implies & p
\end{array}
$$

## 2.3. Disjunctions

$$
\begin{array}{rcl}
p \lor true & \implies & true \\
true \lor p & \implies & true \\
p \lor \neg\, p & \implies & true \\
\neg\, p \lor p & \implies & true \\
p \lor false & \implies & p \\
false \lor p & \implies & p \\
p \lor p & \implies & p
\end{array}
$$

## 2.4. Implications

$$
\begin{array}{rcl}
false \Rightarrow p & \implies & true \\
p \Rightarrow true & \implies & true \\
p \Rightarrow p & \implies & true \\
true \Rightarrow p & \implies & p \\
\neg\, p \Rightarrow p & \implies & p \\
p \Rightarrow false & \implies & \neg\, p \\
p \Rightarrow \neg\, p & \implies & \neg\, p
\end{array}
$$

## 2.5. Equivalences

$$
\begin{array}{rcl}
p \Leftrightarrow p & \Longrightarrow & \mathit{true} \\
p \Leftrightarrow \neg\, p & \Longrightarrow & \mathit{false} \\
\neg\, p \Leftrightarrow p & \Longrightarrow & \mathit{false} \\
\mathit{true} \Leftrightarrow p & \Longrightarrow & p \\
p \Leftrightarrow \mathit{true} & \Longrightarrow & p \\
\mathit{false} \Leftrightarrow p & \Longrightarrow & \neg\, p \\
p \Leftrightarrow \mathit{false} & \Longrightarrow & \neg\, p
\end{array}
$$

## 2.6. Universal quantifications

$$
\begin{array}{rcl}
\forall\, ds \mid \mathit{false} \bullet p & \Longrightarrow & \mathit{true} \\
\forall\, s \bullet \mathit{true} & \Longrightarrow & \mathit{true} \\
\forall\, ds \mid p \bullet p & \Longrightarrow & \mathit{true} \\
\forall \mid \mathit{true} \bullet p & \Longrightarrow & p \\
\forall \mid p_1 \bullet p_2 & \Longrightarrow & p_1 \Rightarrow p_2 \\
\forall\, ds_1 \bullet \forall\, ds_2 \mid p_1 \bullet p_2 & \Longrightarrow & \forall\, ds_1;\ ds_2 \mid p_1 \bullet p_2
\end{array}
$$

where the last rule is applicable only if no name is declared in both $ds_1$ and $ds_2$, and there are no references to $ds_1$ from $ds_2$.

## 2.7. Existential quantifications

$$\begin{array}{rcl}
\exists\, ds \mid \mathit{false} \bullet p & \Longrightarrow & \mathit{false} \\
\exists\, s \bullet \mathit{false} & \Longrightarrow & \mathit{false} \\
\exists \mid \mathit{true} \bullet p & \Longrightarrow & p \\
\exists\, ds \mid p \bullet p & \Longrightarrow & \exists\, ds \bullet p \\
\exists \mid p_1 \bullet p_2 & \Longrightarrow & p_1 \land p_2 \\
\exists\, ds_1 \bullet \exists\, ds_2 \mid p_1 \bullet p_2 & \Longrightarrow & \exists\, ds_1;\ ds_2 \mid p_1 \bullet p_2
\end{array}$$

where the last rule is applicable only if no name is declared in both $ds_1$ and $ds_2$, and there are no references to $ds_1$ from $ds_2$.

## 2.8. Unique existential quantifications

$$\begin{array}{rcl}
\exists_1\, ds \mid \mathit{false} \bullet p & \Longrightarrow & \mathit{false} \\
\exists_1\, s \bullet \mathit{false} & \Longrightarrow & \mathit{false} \\
\exists_1 \mid \mathit{true} \bullet p & \Longrightarrow & p \\
\exists_1\, ds \mid p \bullet p & \Longrightarrow & \exists_1\, ds \bullet p \\
\exists_1 \mid p_1 \bullet p_2 & \Longrightarrow & p_1 \land p_2
\end{array}$$

## 2.9. Relations

| | | |
|---|---|---|
| $e = e$ | $\Longrightarrow$ | *true* |
| $e_1 = e_2$ | $\Longrightarrow$ | *false* where $e_1$ and $e_2$ are distinct literals. |
| $e_1 \in e_2$ | $\Longrightarrow$ | *true* where $e_2$ denotes the carrier set of the non-empty type of $e_1$. |
| $i \in e$ | $\Longrightarrow$ | *true* where $i$'s declaration is of the form $i : e$. |
| $i \in e_2$ | $\Longrightarrow$ | *true* where $i$'s declaration is of the form $i : e$ and $e_2$ is the carrier of $e$. |
| $i.b \in e_b$ | $\Longrightarrow$ | *true* where $i$'s declaration is of the form $i : e_1 \times ... \times e_b \times ... \times e_n$. |
| $i.b \in e_{b2}$ | $\Longrightarrow$ | *true* where $i$'s declaration is of the form $i : e_1 \times ... \times e_b \times ... \times e_n$ a $e_{b2}$ is the carrier set of $e_b$. |
| $e_1.i \in e$ | $\Longrightarrow$ | *true* where $i$'s declaration in the signature of $e_1$ is of the form $i : e$. |
| $e_1.i \in e_2$ | $\Longrightarrow$ | *true* where $i$'s declaration in the signature of $e_1$ is of the form $i : e$ a $e_2$ is the carrier set of $e$. |

Literals are numbers, strings, or extensions (sets, tuples, bindings) containing nothing but literals.

A type is non-empty in the following circumstances. The given type $\mathbb{A}$ is known to be non-empty (this follows from the prelude), and free types that contain at least one element (nullary constructor) are known to be non-empty, but other given types cannot be assumed to be non-empty. Powerset types are known to be non-empty. Cartesian product and schema types are non-empty if all of their components are non-empty. Generic types cannot be assumed to be non-empty. So a type could be empty if it uses any empty given or generic types other than within powerset types.

CADiZ





### 2.10. Schema predicates

$$e \quad \Longrightarrow \quad \mathit{true}$$

where $e$ is a schema predicate, and there is an inclusion declaration $e$ in scope, and the environment of the schema predicate is given entirely by that declaration.

## 3. Expressions

$$\begin{array}{rcl}
\{ds \mid \mathit{false} \bullet e\} & \Longrightarrow & \{\} \\
\{\mid \mathit{true} \bullet e\} & \Longrightarrow & \{e\} \\
\lambda\, ds \mid \mathit{false} \bullet e & \Longrightarrow & \{\} \\
\mu \mid \mathit{true} \bullet e & \Longrightarrow & e \\
\mathrm{let} \mid \mathit{true} \bullet e & \Longrightarrow & e \\
\langle\!|\ i_1 == e_1, \ldots, i_n == e_n\ |\!\rangle . i_k & \Longrightarrow & e_k \\
(e_1, \ldots, e_n).b & \Longrightarrow & e_b \\
\text{if } \mathit{true} \text{ then } e_1 \text{ else } e_2 & \Longrightarrow & e_1 \\
\text{if } \mathit{false} \text{ then } e_1 \text{ else } e_2 & \Longrightarrow & e_2 \\
\text{if } p \text{ then } e \text{ else } e & \Longrightarrow & e \\
\{\langle\!|\ ds\ |\!\rangle\} & \Longrightarrow & [ds] \\
[ds \mid \mathit{false}] & \Longrightarrow & \{\}
\end{array}$$

CADiZ







# 4. Tactic example

$$\text{“}absorption\text{”}\ e\ p$$

This example applies the *absorption* command to expression $e$ and predicate $p$.

*IT 29-Nov-2001*