Isabelle / Isar / HOL proof assistant

Programming und Proving - Kapitel 2 - wesentliche Syntax

Form einer Theorie-Datei T.thy	`theory` T
	`imports` T1 ... Tn
	`begin`
	Definitionen, Theoreme und Beweise
	`end`

Eingebaute Datatypen,Theoremen und Funktionen	siehe Main.pdf
Selbstdefinierte Datatypen	`datatype` domain = DatatypeConstructor1 x | DatatypeConstructor2 y |...
Rekursive funktionen mittels "Pattern Matching": ◦ Priorität von oben nach unten, ◦ (x#xs) ist das Pattern einer Liste, wobei x ist das erstes Element und xs das Rest der Liste, ◦ "_" bedeutet "jedes Beliebige"	`fun` function1 :: "domain ⇒ range" `where` "function1 (DatatypeConstructor1 x) = ..." | "function1 (x#xs) = ..." | "function1 _ = ... " | "funcion1 x = (case x of a ⇒ b | c ⇒ d)" | ...
Nicht rekursive Funktionen	`definition` function2 :: "domain ⇒ range" `where` "function2 x = ..."
Ergebniss der Applikation einer Funktion	`value`(functionName Argument1 Argument2 ...)
Theoreme, Lemmata: ◦ theorem = lemma, ◦ `[simp]:` es geht um eine Gleichung, die zum Vereinfachung von Termen benutzt werden kann. ◦ Definitionen von Funktionen zählen als Theoreme.	`theorem` theorem_name `[simp]:` "bla(bla x) = ..."

Beweis Methoden	`apply(induction x)`
	`apply(induction x rule: function1.induct)`
	`apply(induction x arbitrary: y)`
	`apply(auto)`

Simp wirkt nur auf Subgoal 1	`apply(simp add: theorem1, function1_def,...)`
	`apply(auto simp add: ... simp del: ...)`
	`apply(simp split: Datentyp.split)`

Macht das Thorem ohne Beweis benutzbar	`sorry` `oops`
Schliesst den Beweis ab, und macht das Theorem benutzbar	`done`