Tactic Syntax

/Reference manual

1. Introduction

The syntax of tactics is a CADiℤ-specific extension to the standard Z syntax.

This syntax refers to some non-terminal symbols of the standard syntax, as well as to tokens of the standard lexis and to tokens specific to tactics, all of which are defined elsewhere.

The syntax is formalized below using syntactic metalanguage.

2. Formal definition of tactic syntax

$$Goal \quad = \quad GOAL, [[, NAME, \{, , NAME\},]], Hyp, \vdash ? \ , Concs;$$

$$Hyp \quad = \quad DeclPart, \{\dagger, DeclPart\}, [|, Antes];$$

$$Antes \quad = \quad [Predicate, \{, , Predicate\}];$$

$$Concs \quad = \quad [Predicate, \{, , Predicate\}];$$

$$NamedTacticDef \quad = \quad TACTIC, NAME, TacticDef;$$

$$
\begin{array}{lcl}
\mathit{TacticDef} & = & \mathit{TypedJoker}, \{, , \mathit{TypedJoker}\}, |, \mathit{Tactic}; \\
\\
\mathit{TypedJoker} & = & \mathit{expr}, \mathit{NAME} \\
& | & \mathit{pred}, \mathit{NAME} \\
& | & \mathit{decl}, \mathit{NAME} \\
& | & \mathit{goal}, \mathit{NAME} \\
& | & \mathit{exprs}, \mathit{NAME} \\
& | & \mathit{decls}, \mathit{NAME} \\
& | & \mathit{stxt}, \mathit{NAME} \\
& | & \mathit{name}, \mathit{NAME} \\
& | & \mathit{names}, \mathit{NAME} \\
& | & \mathit{type}, \mathit{NAME} \\
& | & \mathit{str}, \mathit{NAME} \\
& ; &
\end{array}
$$

$$
\begin{array}{rcll}
Tactic & = & RuleName, RuleArg, \{RuleArg\} & \\
& | & Tactic, ;\ , Tactic & (* \ seq \\
& | & (, Tactic, ||, Tactic, \{||, Tactic\},) & (* \ p \\
& | & (, Tactic, !!, Tactic, \{!!, Tactic\},) & (* \\
& | & map, Tactic & (* \\
& | & Tactic, |, Tactic & \\
& | & !, Tactic & \\
& | & skip & \\
& | & fail & \\
& | & rec, NAME, \bullet, Tactic & \\
& | & RecJoker & \\
& | & (, Tactic,) & \\
& | & match, ExprJoker, ExprCase, \{ExprCase\}, ::, . & (\\
& | & match, PredJoker, PredCase, \{PredCase\}, ::, . & \\
& | & match, ExprsJoker, ExprsCase, \{ExprsCase\}, ::, . & (* \\
& | & match, DeclsJoker, DeclsCase, \{DeclsCase\}, ::, . & (* \\
& | & match, StxtJoker, StxtCase, \{StxtCase\}, ::, . & (* \\
& | & match, DeclJoker, DeclCase, \{DeclCase\}, ::, . & (* \\
& | & match, NamesJoker, NamesCase, \{NamesCase\}, ::, . & \\
& | & match, TypeJoker, TypeCase, \{TypeCase\}, ::, . & \\
& | & pat_{ante}, pred, NAME, |, NUMBER, \bullet, Tactic & (* \\
& | & pat_{cons}, pred, NAME, |, NUMBER, \bullet, Tactic & (* \\
& | & pat_{goal}, [TypedJoker, \{, , TypedJoker\}], |, Hyp, \vdash ?\ , Concs, \bullet, Tactic & \\
& | & \mathrm{let}, [TacBinding, \{, , TacBinding\}], \bullet, Tactic & \\
& ; & &
\end{array}
$$

$$ExprCase = ::, [TypedJoker, \{, , TypedJoker\}], |, Expression, \bullet, \{|, Expression, \bullet\}, Tac$$

$$PredCase = ::, [TypedJoker, \{, , TypedJoker\}], |, Predicate, \bullet, \{|, Predicate, \bullet\}, Tactic$$

$$ExprsCase = ::, [TypedJoker, \{, , TypedJoker\}], |, Expressions, \bullet, \{|, Expressions, \bullet\},$$

$$DeclsCase = ::, [TypedJoker, \{, , TypedJoker\}], |, DeclPart, \bullet, \{|, DeclPart, \bullet\}, Tactic$$

$$StxtCase = ::, [TypedJoker, \{, , TypedJoker\}], |, SchemaText, \bullet, \{|, SchemaText, \bullet\}, T$$

$$DeclCase = ::, [TypedJoker, \{, , TypedJoker\}], |, Declaration, \bullet, \{|, Declaration, \bullet\}, Ta$$

$$NamesCase = ::, [TypedJoker, \{, , TypedJoker\}], |, [DeclName, \{, , DeclName\}], \bullet,$$
$$\{|, [DeclName, \{, , DeclName\}], \bullet\}, T$$

$$TypeCase = ::, [TypedJoker, \{, , TypedJoker\}], |, Type, \bullet, \{|, Type, \bullet\}, Tactic;$$

$$RuleName = STRING;$$

$$
\begin{array}{lcl}
\mathit{RuleArg} & = & \mathit{NUMBER} \\
& | & \mathit{STRING} \\
& | & (, \mathit{String}, \{++, \mathit{String}\},) \\
& | & \mathit{ExprJoker} \\
& | & \mathit{PredJoker} \\
& | & \mathit{DeclJoker} \\
& | & \mathit{GoalJoker} \\
& | & \mathit{ExprsJoker} \\
& | & \mathit{DeclsJoker} \\
& | & \mathit{StxtJoker} \\
& | & \mathit{NameJoker} \\
& | & \mathit{NamesJoker} \\
& | & \mathit{TypeJoker} \\
& | & \mathit{StringJoker} \\
& ; &
\end{array}
$$

$$
\begin{array}{rcl}
String & = & STRING \\
& | & ExprJoker \\
& | & PredJoker \\
& | & DeclJoker \\
& | & ExprsJoker \\
& | & DeclsJoker \\
& | & StxtJoker \\
& | & NameJoker \\
& | & NamesJoker \\
& | & TypeJoker \\
& | & StringJoker \\
& ; &
\end{array}
$$

$$TacBinding \quad = \quad [TypedJoker, \{, , TypedJoker\}], ==, TacFunction;$$

$$
\begin{array}{rcl}
TacFunction & = & antecedent, [-], NUMBER \\
& | & consequent, [-], NUMBER \\
& | & parseexpr, STRING \\
& | & parseexprs, STRING \\
& | & parsepred, STRING \\
& | & parsedecl, STRING \\
& | & parsename, STRING \\
& | & parsestxt, STRING \\
& | & declsbefore, DeclsJoker \\
& | & declsafter, DeclsJoker \\
& | & typeof, ExprJoker \\
& | & declof, ExprJoker \\
& ; &
\end{array}
$$

$$
\begin{array}{lcl}
Expression & = & ExprJoker \\
& | & (, ExprJoker,) \\
& | & ExprJoker, as, Expression \\
& | & \_expr \\
& | & all\ Expression\ productions\ of\ the\ standard\ syntax \\
& ; & \\
ExpressionList & = & ExprsJoker \\
& | & ExprsJoker, as, ExpressionList \\
& | & \_exprs \\
& | & all\ ExpressionList\ productions\ of\ the\ standard\ syntax \\
& ; & \\
Predicate & = & PredJoker \\
& | & (, PredJoker,) \\
& | & PredJoker, as, Predicate \\
& | & \_pred \\
& | & all\ Predicate\ productions\ of\ the\ standard\ syntax \\
& ; & \\
SchemaText & = & StxtJoker \\
& | & (, StxtJoker,) \\
& | & StxtJoker, as, SchemaText \\
& | & \_stxt \\
& | & all\ SchemaText\ productions\ of\ the\ standard\ syntax \\
& ; & \\
DeclPart & = & DeclsJoker \\
& | & (, DeclsJoker,) \\
& | & DeclsJoker, as, DeclPart \\
& | & \_decls \\
& | & all\ DeclPart\ productions\ of\ the\ standard\ syntax \\
& . &
\end{array}
$$

Analogously, names jokers can be used in the name list of a hiding expression, along with as patterns and wildcards.

2.1. Operator precedences and associativities

The following table defines the relative precedences of productions and the associativities of their operators. The productions are identified by their rightmost tokens. The rows in the table are ordered so that those entries that bind more weakly appear nearer the top of the table than those that bind more tightly. Those operators that have the same names as Z operators conveniently share the same precedences as those Z operators.

Productions	*Associativity*
as	*right*
• |	*right*
;	*left*
!	*right*

Parentheses may be used to override the default precedences.

3. Notes

GOAL and *TACTIC* are box tokens, distinguishing the two new kinds of paragraph.

The *Goal* syntax recognises all conjectures of the standard syntax, and so subsumes the *Conjecture* syntax.

A *Goal* in a specification may have a name associated with it. The name is written within the mark-up of the paragraph outline, so is not mentioned in this syntax.

TacticDef phrases can appear in text files, in which case the name of the tactic is identified with the name of the file, whereas in specifications the name is given by the *NamedTacticDef* paragraph in which it appears.

The *TypedJoker* notation introduces a new joker, the scope of which is limited to the production that declares it. This condition is enforced by the lexer, as names are mapped to joker tokens through the keyword look-up table.

A *DeclJoker* joker is associated with a single declaration, i.e. either a schema inclusion or the declaration of a single name. Hence in a *DeclCase*, a pattern having multiple names can never match.

The declaration jokers *DeclJoker* and *DeclsJoker* can also be used in those contexts where the standard syntax permits only constant declarations, namely binding extension expressions and subsitution expressions.

The expression list jokers *ExprsJoker* can be used in tuple extensions, set extensions, generic instantiations and sequence arguments. They cannot yet be used in cartesian products, for which a special syntax will be needed.

*RuleArg*s take the place of interactive selections and dialogue responses. The dialogue responses are listed first in order, followed by the selections in the same

order as they would be made interactively. Each selection is represented by a number n, identifying the n'th well-formed formula in the current goal. Each dialogue response is represented by a string. Jokers, as bound to formulae by earlier pattern matching, may be used as selections or dialogue responses. Dialogue response strings may be constructed from the concatenation of smaller strings. A *str* argument is typically used to parametrise a tactic on the name of an arbitrary auxiliary tactic.

General purpose tactics may be written independently of specifications. Where a specification applies a function operator, a tactic can use a juxtaposed function application as a pattern to match against the operator application. Where a specification applies a relation operator, a tactic can use a membership predicate as a pattern to match against the operator application. Where a specification applies a generic operator, a tactic can use a generic instantiation expression as a pattern to match against the operator application.

IT 12-Jan-2002