Tactic Language

Syntax

$$
\begin{array}{rcl}
\mathit{tacexpr} & ::= & \mathit{tacexpr}\ ;\ \mathit{tacexpr} \\
& | & \mathit{tacexpr}\ ||\ \mathit{tacexpr} \\
& | & \texttt{repeat}\ \{\mathit{tacexpr}\} \\
& | & \texttt{try}\ \{\mathit{tacexpr}\} \\
& | & \texttt{rule}\ \mathit{identifier\_op}\texttt{:}\ \mathit{sequent}\ \texttt{if}\ \mathit{sequent\_list\_or\_error} \\
& | & \texttt{call}\ \mathit{identifier} \\
& | & \texttt{without}\ \mathit{formula}\ \{\ \mathit{tacexpr}\ \} \\
& | & \texttt{where}\ \mathit{clause\_list}\ \{\ \mathit{tacexpr}\ \} \\
& | & \texttt{idtac}\ [\ \mathit{message\_token}\ ] \\
& | & \texttt{fail}\ [\ \mathit{message\_token}\ ] \\
& | & \texttt{contradiction}\ \mathit{tacexpr}
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{sequent\_list\_or\_error} & ::= & \mathit{sequent\_list} \\
& | & \texttt{error}\ [\ \mathit{message\_token}\ ]
\end{array}
$$

Semantics

Γ - a goal
Γs - a set of goals
F - a fail
Fs - a set of fails
E - an error
Es a set of errors
Wi - a set of without formulae
Wh - a set of where clauses

SEQUENCE

$$\frac{\begin{array}{c}\mathsf{Wi},\mathsf{Wh} \mid \Gamma, tacexpr_1 \Downarrow \{\Gamma'_1 ... \Gamma'_n\}, \emptyset, \emptyset \\ \mathsf{Wi},\mathsf{Wh} \mid \Gamma'_1, tacexpr_2 \Downarrow \Gamma\mathsf{s}''_1, \mathsf{Fs}''_1, \mathsf{Es}''_1 \\ \vdots \\ \mathsf{Wi},\mathsf{Wh} \mid \Gamma'_n, tacexpr_2 \Downarrow \Gamma\mathsf{s}''_n, \mathsf{Fs}''_n, \mathsf{Es}''_n\end{array}}{\mathsf{Wi},\mathsf{Wh} \mid \Gamma, tacexpr_1; tacexpr_2 \Downarrow \bigcup_i \Gamma\mathsf{s}''_i, \bigcup_i \mathsf{Fs}''_i, \bigcup_i \mathsf{Es}''_i}$$

SEQUENCE FAIL

$$\frac{\begin{array}{c}\mathsf{Wi},\mathsf{Wh} \mid \Gamma, tacexpr_1 \Downarrow \Gamma\mathsf{s}', \mathsf{Fs}', \mathsf{Es}' \\ \mathsf{Fs}' \neq \emptyset \vee \mathsf{Es}' \neq \emptyset\end{array}}{\mathsf{Wi},\mathsf{Wh} \mid \Gamma, tacexpr_1; tacexpr_2 \Downarrow \emptyset, \mathsf{Fs}', \mathsf{Es}'}$$

BRANCHING FAIL

$$\frac{\mathsf{Wi},\mathsf{Wh} \mid \Gamma, tacexpr_1 \Downarrow \Gamma', \mathsf{Fs}', \mathsf{Es}' \qquad \mathsf{Fs}' \neq \emptyset \vee \mathsf{Es}' \neq \emptyset \qquad \mathsf{Wi},\mathsf{Wh} \mid \Gamma, tacexpr_2 \Downarrow \Gamma\mathsf{s}'', \mathsf{Fs}'', \mathsf{Es}''}{\mathsf{Wi},\mathsf{Wh} \mid \Gamma, tacexpr_1 \mid\mid tacexpr_2 \Downarrow \Gamma\mathsf{s}'', \mathsf{Fs}'', \mathsf{Es}' \cup \mathsf{Es}''}$$

BRANCHING

$$\frac{\mathsf{Wi},\mathsf{Wh} \mid \Gamma, tacexpr_1 \Downarrow \Gamma\mathsf{s}', \emptyset, \emptyset}{\mathsf{Wi},\mathsf{Wh} \mid \Gamma, tacexpr_1 \mid\mid tacexpr_2 \Downarrow \Gamma\mathsf{s}', \emptyset, \emptyset}$$

REPEAT CONTINUE

$$\frac{\begin{array}{c}\mathsf{Wi},\mathsf{Wh} \mid \Gamma, tacexpr \Downarrow \{\Gamma'_1 ... \Gamma'_n\}, \emptyset, \emptyset \\ \mathsf{Wi},\mathsf{Wh} \mid \Gamma'_1, \texttt{repeat } \{ \ tacexpr \ \} \Downarrow \Gamma\mathsf{s}''_1, \emptyset, \mathsf{Es}''_1 \\ \vdots \\ \mathsf{Wi},\mathsf{Wh} \mid \Gamma'_n, \texttt{repeat } \{ \ tacexpr \ \} \Downarrow \Gamma\mathsf{s}''_n, \emptyset, \mathsf{Es}''_n\end{array}}{\mathsf{Wi},\mathsf{Wh} \mid \Gamma, \texttt{repeat } \{ \ tacexpr \ \} \Downarrow \bigcup_i \Gamma\mathsf{s}''_i, \emptyset, \bigcup_i \mathsf{Es}''_i}$$

REPEAT STOP

$$\frac{\mathsf{Wi},\mathsf{Wh} \mid \Gamma, tacexpr \Downarrow \Gamma\mathsf{s}', \mathsf{Fs}', \mathsf{Es}' \qquad \mathsf{Fs}' \neq \emptyset \vee \mathsf{Es}' \neq \emptyset}{\mathsf{Wi},\mathsf{Wh} \mid \Gamma, \texttt{repeat } \{ \ tacexpr \ \} \Downarrow \{\Gamma\}, \emptyset, \mathsf{Es}'}$$

TRY SUCCEED

$$\frac{\mathsf{Wi},\mathsf{Wh} \mid \Gamma, tacexpr \Downarrow \Gamma\mathsf{s}', \emptyset, \emptyset}{\mathsf{Wi},\mathsf{Wh} \mid \Gamma, \texttt{try } \{ \ tacexpr \ \} \Downarrow \Gamma\mathsf{s}', \emptyset, \emptyset}$$

TRY FAIL

$$\frac{\mathsf{Wi}, \mathsf{Wh} \mid \Gamma, tacexpr \Downarrow \Gamma\mathsf{s}', \mathsf{Fs}', \mathsf{Es}' \qquad \mathsf{Fs}' \neq \emptyset \vee \mathsf{Es}' \neq \emptyset}{\mathsf{Wi}, \mathsf{Wh} \mid \Gamma, \texttt{try } \{\ tacexpr\ \} \Downarrow \{\Gamma\}, \emptyset, \mathsf{Es}'}$$

RULE

$$\frac{\sigma : Var(sequent) \to terms \qquad \sigma(sequent) = \Gamma \qquad \sigma(\mathsf{Wi}) \notin \Gamma \qquad \sigma(\mathsf{Wh}) \texttt{ holds}}{\mathsf{Wi}, \mathsf{Wh} \mid \Gamma, \texttt{rule } identifier\_op : sequent \texttt{ if } \Gamma\mathsf{s}' \Downarrow \sigma(\Gamma\mathsf{s}'), \emptyset, \emptyset}$$

RULE FAIL

$$\frac{otherwise}{\mathsf{Wi}, \mathsf{Wh} \mid \Gamma, \texttt{rule } id\_op : sequent \texttt{ if } \Gamma\mathsf{s}' \Downarrow \emptyset, \{(\mathsf{F} \texttt{ "failed to apply rule } id\_op\texttt{"})\}, \emptyset}$$

RULE ERROR

$$\mathsf{Wi}, \mathsf{Wh} \mid \Gamma, \texttt{rule } id\_op : sequent \texttt{ if error M with } identifier \Downarrow \emptyset, \emptyset, \{(\mathsf{E}\ \mathsf{M})\}$$

CALL

$$\frac{tacexpr = \texttt{Find\_tactic\_or\_rule\_by\_id}(identifier) \qquad tacexpr \neq null \qquad \mathsf{Wi}, \mathsf{Wh} \mid \Gamma, tacexpr \Downarrow \Gamma\mathsf{s}, \mathsf{Fs}, \mathsf{Es}}{\mathsf{Wi}, \mathsf{Wh} \mid \Gamma, \texttt{call } identifier \Downarrow \Gamma\mathsf{s}, \mathsf{Fs}, \mathsf{Es}}$$

CALL FAIL

$$\frac{null = \texttt{Find\_tactic\_or\_rule\_by\_id}(identifier)}{\mathsf{Wi}, \mathsf{Wh} \mid \Gamma, \texttt{call } identifier \Downarrow \emptyset, \{(\mathsf{F} \texttt{ "failed to find rule } id\_op\texttt{"})\}, \emptyset}$$

WITHOUT

$$\frac{\mathsf{Wi} \cup f, \mathsf{Wh} \mid \Gamma, tacexpr \Downarrow \Gamma\mathsf{s}', \mathsf{Fs}', \mathsf{Es}'}{\mathsf{Wi}, \mathsf{Wh} \mid \Gamma, \texttt{without } f\ \{\ tacexpr\ \} \Downarrow \Gamma\mathsf{s}', \mathsf{Fs}', \mathsf{Es}'}$$

WHERE

$$\frac{\mathsf{Wi}, \mathsf{Wh} \cup clause\_list \mid \Gamma, tacexpr \Downarrow \Gamma\mathsf{s}', \mathsf{Fs}', \mathsf{Es}'}{\mathsf{Wi}, \mathsf{Wh} \mid \Gamma, \texttt{where } clause\_list\ \{\ tacexpr\ \} \Downarrow \Gamma\mathsf{s}', \mathsf{Fs}', \mathsf{Es}'}$$

IDTAC

$\mathsf{Wi}, \mathsf{Wh} \mid \Gamma, \texttt{idtac [message\_token]} \Downarrow \{\Gamma\}, \emptyset, \emptyset$

FAIL

$\mathsf{Wi}, \mathsf{Wh} \mid \Gamma, \texttt{fail [message\_token]} \Downarrow \emptyset, \{(\mathsf{F}\ message\_token)\}, \emptyset$

CONTRADICTION

$$\frac{\Gamma = \Sigma \vdash \Sigma' \qquad \mathsf{Wi}, \mathsf{Wh} \mid (\Sigma \vdash False),\ tacexpr \Downarrow \Gamma\mathsf{s}, \mathsf{Fs}, \mathsf{Es}}{\mathsf{Wi}, \mathsf{Wh} \mid \Gamma, \texttt{contradiction}\ tacexpr \Downarrow \Gamma\mathsf{s}, \mathsf{Fs}, \mathsf{Es}}$$

Rule

- **constructor** identifier
- **import** STRING_CONSTANT **;**
- **rule** *identifier_op* **:** *sequent without where* **if** *sequent_list_or_list*
- **rewrite** *identifier_op* **:** identifier **(** jargument_list **)** = jargument *if-clause without_simp where*
- **rewrite** *identifier_op* ***:** identifier **(** jargument_list **)** = jargument *if-clause without_simp where*
- **abstraction** *identifier_op* **:** *formula* $\rightsquigarrow$ *formula where*
- **equiv** *identifier_op* **:** *formula* $-*$ *formula* $\Leftrightarrow$ *formula without_simp*
- **equiv** *identifier_op* **:** *formula* $\Rightarrow$ *formula* $\Leftrightarrow$ *formula without_simp*
- **equiv** *identifier_op* **:** *formula* $\Rightarrow$ *formula without_simp*
- **equiv** *identifier_op* **:** *formula* $\Leftrightarrow$ *formula without_simp*

identifier_op

- /*empty*/
- identifier

sequent

- *spatial_list* | *formula* $\vdash$ *formula*
- *spatial_list* | *formula* $\vdash$ *formula* $\dashv$ *formula*
- /* used because the collapse form is || which is a reserved token */ *spatial_list* || *formula* $\vdash$ *formula*

spatial_list

- *spacial_list_ne*
- /* empty */

spatial_list_ne

- *spacial_at* * spacial_list_ne
- *spatial_at*

spatial_at

- jargument.field_signature ↦ jargument
- identifier (jargument_list)

formula

- /*empty*/
- **Emp**
- **False**
- **Garbage**
- lvariable.jargument ↦ jargument
- **!** identifier (jargument_list)
- identifier (jargument_list)
- full_identifier (jargument_list)
- formula ***** formula
- formula | formula

- formula ‖ formula
- lvariable **:** identifier
- jargument binop_cmp jargument
- jargument = jargument
- (formula)

without

- **without** *formula*
- **without** *formula* ⊢ *formula*

where

- **where** *clause_list*
- /*empty*/

clause_list

- *clause*
- *clause* **;** clause_list

clause

- varterm **notincontext**
- varterm **notin** jargument
- *formula* **puregard**

sequent_list_or_list

- *sequent_list*
- *sequent_list* **;** sequent_list_or_list

sequent_list

- **True**
- *sequent*
- *sequent* **;** sequent_list

if_clause

- **if** *formula*

without_simp

- **without** *formula*
- /*empty*/