Constructor-based Inductive Theorem Prover List of Commands

Daniel Găină

Institute of Mathematics for Industry, Kyushu University
daniel@imi.kyushu-u.ac.jp

Abstract. Constructor-based Theorem Prover (CITP) is a tool for proving inductive properties of software systems specified with constructor-based logics. The present document describes the main commands supported by CITP.

1 Introduction

A goal $< \mathtt{M}, \mathtt{SS} >$ consists of a module/specification `M` and a set of sentences `SS` rather than a single formula like in Coq [1] or PVS [4] approach. We may use the notation $\mathtt{M} \vdash \mathtt{SS}$ to represent a goal. The proof rules

$$\frac{< \mathtt{M}_1, \mathtt{SS}_1 > \ldots < \mathtt{M}_n, \mathtt{SS}_n >}{< \mathtt{M}, \mathtt{SS} >}$$

can be regarded, upside down, as basic tactics for decomposing problems. The syntax for entering a goal is

$$(\mathtt{goal}\ \mathtt{M} \vdash \mathtt{SS})$$

where `M` is the name of a Maude program representing a specification, and `SS` is a set of sentences. Note that a sentence can be an equation, a membership axiom or a so-called rule. After entering the goal, the user needs to discharge it by giving commands to the tool. The commands available for CITP are described in the next section.

2 User interaction

CITP consists of two parts:

(1) the core, which implements the proof tactics, and
(2) the user interface, which implements the input parser, displays the results, and defines commands to interact with the users.

The proof tactics are functions `GoalList` $\rightarrow$ `GoalList` defined on lists of goals with values in lists of goals. The *basic proof tactics* are obtained from proof rules, which are functions `Goal` $\rightarrow$ `GoalList` that take goals as arguments and return lists of goals. Given a proof rule `pr` : `Goal` $\rightarrow$ `GoalList` then the proof tactic `pr` : `GoalList` $\rightarrow$ `GoalList` associated to `pr` : `Goal` $\rightarrow$ `GoalList` is defined by $\mathtt{pr}(\mathtt{G}_1 \ldots \mathtt{G}_n) = \mathtt{pr}(\mathtt{G}_1) \ldots \mathtt{pr}(\mathtt{G}_n)$. It is worth noting that are proof tactics which are not derived from proof rules such as `select` (see the next subsection).

2.1 General proof tactics

The general proof tactics are sound for all specifications and they can be divided into two subcategories:

(1) The general proof tactics derived from the proof rules of the specification calculus defined in [2] and refined for applications in [3].

Tactic	Abbrev.
Simultaneous induction	`ind on`
Case Analysis	`ca`
Theorem of Constants	`tc`
Implication	`imp`
Reduction	`red`
Initialisation	`init`
Critical pairs left	`cp-l`
Critical pairs right	`cp-r`

In order to apply simultaneous induction to a goal the user must specify the induction variables.

$$(\texttt{ind on X}_1 : \texttt{S}_1 \ldots \texttt{X}_n : \texttt{S}_n)$$

where $\texttt{X}_i$ is the name of the variable and $\texttt{S}_i$ is its sort. Note that the induction can be performed on a set of variables
None-executable equations and rules can be instantiated during the proof process by substituting terms for variables.

$$(\texttt{init E by X}_1 : \texttt{S}_1 \leftarrow \texttt{T}_1; \ldots; \texttt{X}_n : \texttt{S}_n \leftarrow \texttt{T}_n)$$

Note that `E` can be wither an equation or a rule and it can be also referred by its name given with `metadata` attribute.
CITP allows to join critical pairs by applying the commands `cp-l` and `cp-r`.

```
(cp-l eq T1 = T2 ; eq T1' = T2' ;)
```

The result of giving the above command to a goal $< \texttt{M}, \texttt{SS} >$ is another goal $< \texttt{M}', \texttt{SS} >$, where $\texttt{M}'$ is obtained from `M` by joining (from left to right) a critical pair that resulted from unifying a subterm of `T1` with `T2`. The following command

```
(cp-l eq T1 = T2 ; eq T1' = T2' ; skip N)
```

is similar to `(cp-l eq T1 = T2 ; eq T1' = T2' ;)` but it skips `N` solutions when trying to unify a subterm of `T1` with `T2`. The tactic `cp-r` is similar to `cp-l` but the joining is performed from right to left.

(2) The general proof tactics that manage the proof processes.

Tactic	Abbrev.
Select	`select`
Dot	`.`

The current goal is the last goal in the list and the following command

```
(select N)
```

moves the `N` goal to the bottom of the list.
All proof tactics are applicable to the entire list of goals. If the user wishes to apply the proof tactics only to the current goal then the command dot should be used. For example

```
(. tc red)
```

applies theorem of constants and reduction to the current goal.

Remark 1. Note that the tactics `Select` and `Dot` are not basic tactics as they are not derived from proof rules.

2.2 Specific proof tactics

The specific proof tactics are sound for initial data types that are often used in applications such as sequences/lists, sets and pairs as long as they are protected.

Tactic	Abbrev.
Induction based on membership axioms	`indx`
Case analysis for sequences and sets	`cs`

Remark 2. The following tactics are designed for goals consisting of a specification and a single formula: `ca, tc, imp, red, cs, pair`. However, if one of these tactics is applied to a goal of the form $\mathtt{M} \vdash \{\mathtt{E}_1, \ldots, \mathtt{E}_n\}$, the goal is decomposed into a list of subgoals $(\mathtt{M} \vdash \mathtt{E}_1), \ldots, (\mathtt{M} \vdash \mathtt{E}_n)$ and then the tactic is applied to each goal $\mathtt{M} \vdash \mathtt{E}_i$.

2.3 Proof tactic lists

Any combination of the proof tactics above is also valid for CITP. For example, the following

```
(goal M ⊢ E ind var X : S ind tc red)
```

is a valid command, where `M` is a name of a module in Maude and `E` is an equation, or a membership axiom or a rule.

2.4 Commands

The commands available for CITP do not consist only of lists of proof tactics but also of commands that do not modify the current goal list, or, in some cases, it is necessary to return to a previous state of the proof process due to erroneous applications of proof tactics.

Command	Abbrev .
Show proof	`show proof`
Show goals	`show goals`
Show current goal	`show current goal`
Rollback	`rollback`
Reduce term	`redTerm`

References

1. Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. Springer Verlag, 2004.
2. K. Futatsugi, D. Găină, and K. Ogata. Principles of proof scores in CafeOBJ. *Theoretical Computer Science*, 464:90–112, 2012.
3. D. Găină, D. Lucanu, K. Ogata, and K. Futatsugi. On Automation of OTS/CafeOBJ method. 2013. submitted.
4. S. Owre, J. M. Rushby, , and N. Shankar. PVS: A prototype verification system. In D. Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga, NY, jun 1992. Springer-Verlag.