A Formalism for the Control Component of OMRS: NQTHM as a Case Study

Alessandro Coglio

1 Introduction

The control component of OMRS [6] must specify the strategies according to which systems manipulate reasoning structures (see [6]) in order to perform deductions (e.g.: add this sequent node, then add this link node connecting it to these sequent nodes; now if a certain condition is verified then add this sequent node else add this other one; and so on).

In §2 we review what in literature addresses the problem of specifying strategies of manipulation of proof trees to perform deductions[1], arguing for the inadequacy of existing formalisms to specify complex strategies of manipulations of reasoning structures in a practical way. In §3 we thus propose a new formalism for the control component of OMRS, and in §4 we describe its application to the specification of the control strategies of NQTHM [1, 2]. Finally, in §5 we describe future work to be done.

2 Control in Literature

In literature the most interesting formalism addressing the problem of specifying strategies of manipulation of proof trees to perform deductions is the one of tactics and tacticals. There are basically two flavors of tactics and tacticals, which in NuPRL [4] terminology are called 'refinement tactics' (with related tacticals) and 'transformation tactics' (with related tacticals).

A *refinement tactic*[2] is basically[3] a procedure which, when called upon a sequent[4], can terminate returning a (possibly empty) finite sequence of sequents (in such case we say the tactic 'succeeds'), terminate returning `fail` (in such case we say the tactic 'fails'), or not terminate at all. If it succeeds, the returned sequents are such that the input sequent is derivable from them (in other words, the problem of proving the input sequent has been reduced to the supposedly simpler problems of proving the returned sequents), so that we can view the execution of the tactic as the construction of a proof tree whose root is the input sequent and whose leaves are the returned sequents; if it fails, the tactic has not been able to decompose the problem of proving the input sequent into sub-problems.

[1]This problem is closely related to ours, since reasoning structures are a "generalization" of proof trees.

[2]Refinement tactics are used in Edinburgh LCF [7], GETFOL [5] and NuPRL [4].

[3]We give here a simplified description of refinement tactics (in particular, we neglect the notion of validation; see [7] for further information), which is adequate to our present discussion.

[4]We use reasoning theories terminology (see [6]).

Tacticals are functions which are applied to tactics to yield more complex tactics: tacticals allow us to invoke a certain tactic and if it fails invoke an alternative tactic, or to invoke a certain tactic and then (possibly recursively) invoke other tactics upon the sequents returned by the first one. Usually one starts with so-called 'primitive tactics', which correspond to backward applications of inference rules (primitive tactics fail if the corresponding inference rules are not applicable, otherwise they succeed and their executions can be viewed as constructions of proof trees of depth 1 in the obvious way), then builds more complex tactics by means of tacticals, and the executions of these more complex tactics can be viewed, in case of success, as step-by-step constructions of proof trees (e.g. when a tactic is invoked and then other tactics are invoked upon the sequents returned by the first one, the trees constructed by the other tactics are "attached" as sub-trees of the tree constructed by the first tactic): it is thus possible to specify strategies of applications of inference rules to construct proof trees.

While on one hand refinement tactics are formally very clear, on the other hand they are not very powerful, that is they cannot be used in a practical way to express complex manipulations of proof trees: in particular, there is no way to deal with schematic reasoning (i.e. having proof trees with schematic variables which are substituted by expressions as deduction proceeds, and with schematic constraints whose satisfaction or satisfiability is checked after their schematic variables are substituted).

A *transformation tactic*[5] is basically[6] a procedure which, when called upon a proof tree, can terminate returning another proof tree (in such case we say the tactic 'succeeds'), terminate returning `fail` (in such case we say the tactic 'fails'), or not terminate at all: if it succeeds, it really performs a transformation (hence the name) of the input proof tree into the output one. By means of tacticals which are analogous to those for refinement tactics it is possible to build more complex transformation tactics starting with simple ones. Refinement tactics can be viewed as a particular case of transformation tactics. Transformation tactics can specify arbitrary manipulations of proof trees; anyway, they are "too general" and hence not formally very clear. Furthermore, since transformation tactics (as well as refinement tactics) only "deal with" sequents and inference rules, they can only specify computations over logical information, while complex systems also perform computations over control (i.e. non-logical) information (e.g. how a certain sequent has been produced starting with others).

3 The Proposed Formalism

The formalism we propose (described in detail in [3]) consists of two somehow orthogonal parts: on one hand, we provide a way of "augmenting" a given reasoning theory with control information in a clear way, thus having rules which express elaborations of both logical and control information; on the other hand, we define a new class of tactics and related tacticals which, while being much more powerful than refinement tactics, are still formally very clear (unlike

[5]Transformation tactics are used in NuPRL.

[6]We give here a simplified description of transformation tactics (see [4] for further information), which is adequate to our present discussion.

transformation tactics).

A reasoning theory (see [6]) basically consists of sequents (representing assertions), constraints (representing restrictions), instantiations (representing substitutions of schematic parts), and inference rules (representing elaborations over sequents); sequents and constraints can be schematic, and instantiations can be applied to them thus performing substitutions of schematic parts. Given a reasoning theory, we "augment" it with control information in the following way. We define *annotated sequents* (representing assertions over both logical and control information), *annotated constraints* (representing restrictions over both logical and control information), *annotated instantiations* (representing substitutions of both logical and control information), and also a *control-logic mapping*, that is a function mapping annotated sequents, annotated constraints, and annotated instantiations to sequents, constraints, and instantiations (of the given reasoning theory), respectively, in such a way that the application of the control-logic mapping to annotated entities achieves the effect of "eliminating" control information from them, leaving logical information intact. Furthermore, we define *control-inference rules* (representing elaborations over both logical and control information contained in annotated sequents) and *control rules* (representing elaborations over only control information contained in annotated sequents), which are exactly inference rules over annotated sequents which are "logically justified" in the following sense: for each rule instance of each control-inference rule, by applying the control-logic mapping to its premises, conclusion and applicability conditions, we have that the "non-annotated conclusion" is derivable form the "non-annotated premises" provided the "non-annotated applicability conditions" are satisfied; each rule instance of a control rule has exactly one premise, and by applying the control-logic mapping to the premise and the conclusion we obtain the same non-annotated sequent. Such annotated sequents, annotated constraints, annotated instantiations, control-inference rules, control rules, and control-logic mapping constitute a *control reasoning theory* over the given reasoning theory.

The new tactics are procedures which, given a sequent as input, may terminate returning an instantiation, a (possibly empty) finite sequence of sequents, and a (possibly empty) finite set of constraints (in such case we say the tactic 'succeeds'), terminate returning a failure[7] (in such case we say the tactic 'fails'), or not terminate at all: if it succeeds, the meaning of the returned instantiation, sequents and constraints is that the sequent obtained applying the returned instantiation to the input sequent is derivable from the returned sequents provided the returned constraints are satisfied, so that we can view the execution of the tactic as the construction of a tree-shaped reasoning structure whose "root" is the instance of the input sequent, and whose "leaves" are the output sequents. We have two new tacticals (which consist in "generalizations" of the ones for refinement and transformation tactics): tactical `ORELSE` is used to invoke a tactic, and if it fails to invoke different alternative tactics according to the returned failure; tactical `ANDTHEN` is used to invoke a tactic and then invoke other tactics upon the sequents returned by the first one, with returned instantiations being applied to sequents and constraints in such a way that information is "moved around" as instantiations, and with constraints which can

[7]Unlike both refinement and transformation tactics, which always return the same value `fail` in case of failure, in our formalism tactics can return different values (called 'failures') according to the cause of failure, thus giving some information about why the tactic failed.

be checked for satisfaction or satisfiability after instantiating them (constraints are dealt with by *constraint solvers*, that is procedures which, called upon a finite set of constraints, either return another finite set of constraints whose satisfaction implies the satisfaction of the input constraints, or return a failure).

Putting things together, given a reasoning theory one usually acts in the following way: first a control reasoning theory over the given reasoning theory is defined, then starting with 'primitive tactics' associated to control-inference rules and control rules (in fact, if we "forget" the control-logic mapping, a control reasoning theory is exactly a reasoning theory, thus tactics can deal with annotated sequents, constraints and instantiations as well as non-annotated ones) complex tactics are built by means of tacticals `ORELSE` and `ANDTHEN` (possibly defining some tactics recursively), which specify in a formally clear way strategies of applications of control-inference rules and control rules to annotated sequents to construct tree-shaped control reasoning structures (which can be straightforwardly mapped to tree-shaped reasoning structures), similarly to refinement tactics but much more powerfully, because it is possible to deal with schematic reasoning.

4 NQTHM as a Case Study

The proposed formalism has been successfully applied (see [3]) to the specification of the complex strategies of NQTHM, the state-of-the-art Boyer-Moore theorem prover [1, 2]; only the global strategy (which calls various inference processes) and the strategy of the simplifier (one of the inference processes) have been specified, the other processes having been specified as black boxes by means of appropriate functions (anyway the global strategy and the one of the simplifier are the most complex ones of the system).

Just to give the flavor of what has been done, here is an example of inference rule of the reasoning theory which has been defined to specify the logic component of NQTHM as an OMRS (where cl is a clause, $\widetilde{cl}$ is a finite set of clauses, and $\widehat{cl}$ is a finite multi-set of clauses):

$$\mathsf{CallSimp} \quad \frac{evh \vdash_{\mathrm{SIMP}} cl \rightsquigarrow \widetilde{cl} \qquad evh \vdash_{\mathrm{W}} \widehat{cl} \cup \widetilde{cl}}{evh \vdash_{\mathrm{W}} \widehat{cl} \cup \{cl\}} \quad \left\{ \; \widetilde{cl} \neq \{cl\} \; \right\}$$

This inference rule represents, from a logical point of view, the call of the simplifier by the global strategy: if clause cl simplifies to clauses $\widetilde{cl}$ (and if $\widetilde{cl}$ is not the singleton set containing cl, as expressed by the constraint), we replace cl with $\widetilde{cl}$ in the multi-set of clauses which must be proved.

And here is an example of a control-inference rule, which is logically justified by the inference rule above:

$$\mathsf{CallSimp}$$
$$\frac{\begin{array}{l} evh \vdash_{\mathrm{SIMP}} cl, hst; \widetilde{iht}\,, \widetilde{ict} \rightsquigarrow \overline{cl}\,', hste \\ evh \vdash_{\mathrm{W}} \overline{cl}\,' \diamond \overline{cl}\,, [[\langle \mathrm{S}, cl, hste\rangle] \diamond hst]^{|\overline{cl}\,'|} \diamond \overline{hst}\,; \overline{\overline{cl}}\,, \overline{tag}\,; \widetilde{iht}\,, \widetilde{ict} \end{array}}{evh \vdash_{\mathrm{W}} [cl] \diamond \overline{cl}\,, [hst] \diamond \overline{hst}\,; \overline{\overline{cl}}\,, \overline{tag}\,; \widetilde{iht}\,, \widetilde{ict}}$$
$$\left\{ \; \overline{cl}\,' \neq [cl] \; \right\}$$

This control-inference rule represents, from both a logical and a control point of view, the call of the simplifier by the global strategy: clauses to be proved are ordered in sequences, and if the first clause *cl* in the sequence simplifies to clauses $\overline{cl}\,'$ (and if $\overline{cl}\,'$ is not the singleton sequence containing *cl*, as expressed by the constraint) then *cl* is replaced with $\overline{cl}\,'$ in the sequence ($\diamond$ is the sequence concatenation operator); *hst* and $\overline{hst}$ are respectively a history and a finite sequence of histories (in NQTHM clauses are accompanied by the histories of how they have been produced), and the histories of the newly produced clauses $\overline{cl}\,'$ are obtained from the history *hst* of *cl* by adding the information S (identifying the simplifier), *cl* (the starting clause), and *hste* (produced by the simplifier).

Primitive tactic `CallSimpTac` associated to control-inference rule CallSimp is used, together with other primitive tactics, to define complex tactics in a bottom-up fashion by means of tacticals `ORELSE` and `ANDTHEN`: the whole strategy of NQTHM is specified by complex tactic `NqthmTac`.

5 Future Work

Our new tactics (and tacticals) are formally very clear and much more powerful than refinement tactics; anyway, as future work we would like to define even more powerful tactics, which for example are able to specify constructions of (control) reasoning structures not limited to be tree-shaped.

References

[1] R.S. Boyer and J.S. Moore. *A Computational Logic.* Academic Press, 1979. ACM monograph series.

[2] R.S. Boyer and J.S. Moore. *A Computational Logic Handbook.* Academic Press, 1988.

[3] A. Coglio. “Definizione di un formalismo per la specifica delle strategie di inferenza dei sistemi di ragionamento meccanizzato e sua applicazione ad un sistema allo stato dell’arte”, 1996. Thesis, DIST - University of Genoa (Italy).

[4] R.L. Constable, S.F. Allen, H.M. Bromley, et al. *Implementing Mathematics with the NuPRL Proof Development System.* Prentice Hall, 1986.

[5] F. Giunchiglia. `GETFOL` Manual - `GETFOL` version 2.0. Technical Report 92-0010, DIST - University of Genoa, Genoa, Italy, March 1994.

[6] F. Giunchiglia, P. Pecchiari, and C. Talcott. Reasoning Theories: Towards an Architecture for Open Mechanized Reasoning Systems. Technical Report 9409-15, IRST, Trento, Italy, 1994. Also DIST Technical Report 94-0021, DIST, University of Genova, Italy. Also published as Stanford Computer Science Department Technical note number STAN-CS-TN-94-15, Stanford University.

[7] M.J. Gordon, A.J. Milner, and C.P. Wadsworth. *Edinburgh LCF - A mechanized logic of computation*, volume 78 of *Lecture Notes in Computer Science.* Springer Verlag, 1979.