Tactic-based theorem proving in first-order modal and temporal logics

School of Informatics, University of Edinburgh

Centre for Intelligent Systems and their Applications

Tactic-based theorem proving in first-order modal and temporal logics

by

Claudio Castellini, Alan Smaill

Informatics Research Report EDI-INF-RR-0155

School of Informatics **June 2001**
http://www.informatics.ed.ac.uk/

Tactic-based theorem proving in first-order modal and temporal logics

Claudio Castellini, Alan Smaill

Informatics Research Report EDI-INF-RR-0155

SCHOOL *of* INFORMATICS
Centre for Intelligent Systems and their Applications

June 2001

appears in Proceedings of IJCAR 2001, WS 10

Abstract :

We describe the ongoing work on a tactic-based theorem prover for First-Order Modal and Temporal Logics (FOTLs for the temporal ones). In formal methods, especially temporal logics play a determining role; in particular, FOTLs are natural whenever the modeled systems are infinite-state. But reasoning in FOTLs is hard and few approaches have so far proved effective. Here we introduce a family of sequent calculi for first-order modal and temporal logics which is modular in the structure of time; moreover, we present a tactic-based modal/temporal theorem prover enforcing this approach, obtained employing the higher-order logic programming language lambda-Prolog. Finally, we show some promising experimental results and raise some open issues. We believe that, together with the Proof Planning approach, our system will eventually be able to improve the state of the art of formal methods through the use of FOTLs.

Keywords : theorem proving, modal logics, temporal logics

Tactic-based theorem proving in First-Order Modal and Temporal Logics

Claudio Castellini and Alan Smaill

Division of Informatics
University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, UK

Abstract. We describe the ongoing work on a tactic-based theorem prover for First-Order Modal and Temporal Logics (FOTLs for the temporal ones). In formal methods, especially temporal logics play a determining role; in particular, FOTLs are natural whenever the modeled systems are infinite-state. But reasoning in FOTLs is hard and few approaches have so far proved effective. Here we introduce a family of sequent calculi for first-order modal and temporal logics which is modular in the structure of time; moreover, we present a tactic-based modal/temporal theorem prover enforcing this approach, obtained employing the higher-order logic programming language λProlog. Finally, we show some promising experimental results and raise some open issues. We believe that, together with the Proof Planning approach, our system will eventually be able to improve the state of the art of formal methods through the use of FOTLs.

1 Introduction

Temporal logics are extensions of classical logic, dealing with the concept of time and how properties of a dynamic system change through time [GHR93]. From another point of view, temporal logics are modal logics whose Kripke frame enforces the intended structure of time. Temporal logics are used in specification and verification of multi-agent, concurrent and reactive systems, programs, circuits and protocols (see e.g. [MP92]). Therefore, effective temporal thorem proving is highly desirable.

While several effective approaches have been developed for propositional modal and temporal logics, the situation is quite hard when we switch to first-order temporal logics (FOTLs), which constitute a richer class of languages, mostly undecidable and nevertheless natural for the specification of infinite-state systems. Even apparently small fragments of FOTL over the natural numbers, for instance, are not recursively enumerable (see e.g. [BO95] and references therein).

This paper describes the work we are carrying on about first-order modal and temporal theorem proving: we give a labelled, modular presentation of first-order modal and temporal logics, setting up a family of sequent calculi which allows different Kripke structure / time structures to be dealt with; then we describe a modular, tactic-based modal/temporal theorem prover which supports the

approach described above. The implementation closely follows Amy Felty's work on higher-order, tactic-based theorem proving in λProlog [NM98,Fel93].

The choice of a modular approach to FOTLs in a labelled modal logic style is motivated by three considerations: *(i)* the temporal logics we are interested in are modal logics whose Kripke frame enforces the structure of time (linear, branching, discrete, continuous, etc.); *(ii)* labelled deduction in modal logics, even in the first-order case, is feasible (see, e.g., [BMV98]); *(iii)* most FOTLs share the same syntax and only differ in the underlying structure of time.

The paper is structured as follows: Section 2 describes the family of sequent calculi $\mathcal{TL}_{\mathbf{L}}$; Section 3 describes the theorem prover FTL and shows some experimental results; Section 4 outlines our future work.

2 A modular presentation of modal and temporal logics

The language we use for our sequent calculi must be flexible enough for a wide family of modal and temporal logic; at the same time, it must be easily manageable by a small set of sequent rules. So we choose to embed modal and temporal logics in a *labelled deductive system* whose basic unit of information is the *labelled formula* rather than a simple formula. We also use formulae called *constraints* to express the relations that must hold between labels. A thorough exposition of labelled deduction can be found in [Gab96].

In our setting, a labelled formula has the shape $\varphi@\tau$ where φ is a *logical formula* and τ is a *label*. The intuitive meaning is: "at the (world, situation, instant) denoted by τ, φ holds". Logical formulae are standard formulae of a full first-order language; they are combinations of atoms / logical formulae via standard first-order connectives ($\neg$, $\supset$, $\forall$) plus the unary modal connective $\Box$ (other connectives $\wedge$, $\vee$, $\leftrightarrow$, $\exists$ and $\Diamond$ may be defined from these). Labels are either variable symbols τ or the constant 0. A *constraint* is the application of the binary relation $\preceq$ to two labels.

The language of labels / constraints is disjoint from that of logical formulae; this way, it can be augmented to fit the requirements of the logic we are examining. For example, in the following we will introduce a "successor" function σ into the language of labels, thus stepping towards a logic based on discrete frames.

Truth of labelled formulae and constraints is defined in a quite standard way (see [AM90,BD92]). Let

$$\mathbf{M} = \langle \mathcal{W}, R, \mathcal{D}, I \rangle$$

be a tuple where $\mathcal{W}$ is the set of possible worlds, $R \subseteq \mathcal{W} \times \mathcal{W}$ is the accessibility relation, $\mathcal{D}$ is the domain of quantification and I maps a world and a predicate symbol to a predicate over $\mathcal{D}$. We deal with quantified modal logics on *constant domains*, that is, the domain of quantification $\mathcal{D}$ is the same in all possible worlds. Moreover, we employ *rigid designators*, that is, the only dynamic objects are predicates.

We also need a first-order interpretation I_l mapping $\preceq$ to R, 0 to an element of $\mathcal{W}$, and variable symbols to elements in $\mathcal{W}$. We indicate possible worlds with

the letter w and intend that $w, w_i, w', \ldots$ are the objects referred to by labels $\tau, \tau_i, \tau', \ldots$

Let α be a function recursively mapping logical terms to values in $\mathcal{D}$ (assignment); the notion of a formula being true in $\mathbf{M}$ under α, written $\mathbf{M}, \alpha \models \varphi$, is recursively defined as follows:

$$
\begin{array}{lll}
\mathbf{M}, \alpha \models \tau_1 \preceq \tau_2 & \text{iff} & (w_1, w_2) \in R \\
\mathbf{M}, \alpha \models p(s_1, \ldots, s_n)@\tau & \text{iff} & (\alpha(s_1), \ldots, \alpha(s_n)) \in I(w, p) \\
\mathbf{M}, \alpha \models \neg\varphi@\tau & \text{iff} & \mathbf{M}, \alpha \not\models \varphi@\tau \\
\mathbf{M}, \alpha \models \varphi \supset \psi@\tau & \text{iff} & \mathbf{M}, \alpha \not\models \varphi@\tau \text{ or } \mathbf{M}, \alpha \models \psi@\tau \\
\mathbf{M}, \alpha \models \forall x.\varphi@\tau & \text{iff} & \text{for all } d \in \mathcal{D}, \\
 & & \quad \mathbf{M}, \alpha^{[d/x]} \models \varphi[d/x]@\tau \\
\mathbf{M}, \alpha \models \Box\varphi@\tau & \text{iff} & \text{for all } w' \in \mathcal{W}, \\
 & & \quad \mathbf{M}, \alpha \not\models \tau \preceq \tau' \text{ or } \mathbf{M}, \alpha \models \varphi@\tau'
\end{array}
$$

Given a standard notion of sequents as set of formulae (as presented, e.g., in [Wal89]), we introduce the sequent calculus $\mathcal{TL}_{\mathbf{K}}$ for quantified $\mathbf{K}$ on constant domains and with rigid designators in Figure 1.

$$
\frac{}{\Gamma, \varphi@\tau \longrightarrow \varphi@\tau, \Delta}\ \mathsf{ax} \qquad \frac{\Gamma \vdash_{\mathcal{A}} \Delta}{\Gamma \longrightarrow \Delta}\ \mathsf{ent}
$$

$$
\frac{\Gamma \longrightarrow \varphi@\tau, \Delta}{\Gamma, \neg\varphi@\tau \longrightarrow \Delta}\ l\neg \qquad \frac{\Gamma, \varphi@\tau \longrightarrow \Delta}{\Gamma \longrightarrow \neg\varphi@\tau, \Delta}\ r\neg
$$

$$
\frac{\Gamma \longrightarrow \varphi@\tau, \Delta \quad \Gamma, \psi@\tau \longrightarrow \Delta}{\Gamma, \varphi \supset \psi@\tau \longrightarrow \Delta}\ l\supset \qquad \frac{\Gamma, \varphi@\tau \longrightarrow \psi@\tau, \Delta}{\Gamma \longrightarrow \varphi \supset \psi@\tau, \Delta}\ r\supset
$$

$$
\frac{\Gamma, \varphi[c/x]@\tau \longrightarrow \Delta}{\Gamma, \forall x.\varphi@\tau \longrightarrow \Delta}\ l\forall \qquad \frac{\Gamma \longrightarrow \varphi[a/x]@\tau, \Delta}{\Gamma \longrightarrow \forall x.\varphi@\tau, \Delta}\ r\forall
$$

$$
\frac{\Gamma, \varphi@t \longrightarrow \Delta \quad \Gamma \longrightarrow \tau \preceq t, \Delta}{\Gamma, \Box\varphi@\tau \longrightarrow \Delta}\ l\Box \qquad \frac{\Gamma, \tau \preceq t_a \longrightarrow \varphi@t_a, \Delta}{\Gamma \longrightarrow \Box\varphi@\tau, \Delta}\ r\Box
$$

Fig. 1. the calculus $\mathcal{TL}_{\mathbf{K}}$ for quantified $\mathbf{K}$. a and t_a cannot appear in the conclusion of $r\forall$ and $r\Box$.

The entailment rule ent allows to close a branch if the constraints in Γ "entail" at least one constraint among the consequents; the entailment relation $\vdash_{\mathcal{A}}$ represents first-order deduction from a set $\mathcal{A}$ of axioms which enforce the desired properties of the frame. In this case $\mathcal{A}$ is empty and ent reduces to an axiomatic rule for constraints:

$$\frac{}{\Gamma, \tau_1 \preceq \tau_2 \longrightarrow \tau_1 \preceq \tau_2, \Delta}\ \mathsf{ax_c}$$

It is possible to prove in $\mathcal{TL}_{\mathbf{K}}$ a number of characteristic axioms of **K** (e.g., modal modus ponens and both Barcan formulae). Also, the rule of necessitation is naturally enforced.

From the theory of correspondence [van84] we know that most useful properties of Kripke frames, which also characterize modal logics *in toto*, are expressible as modal axioms and have first-order formulations. So we extend $\mathcal{TL}_{\mathbf{K}}$ with new sequent rules corresponding to first-order conditions on $\preceq$. For instance, **T** is characterised by reflexive frames; so we add rule refl to $\mathcal{TL}_{\mathbf{K}}$ and call the resulting calculus $\mathcal{TL}_{\mathbf{T}}$:

$$\frac{\Gamma, \tau \preceq \tau \longrightarrow \Delta}{\Gamma \longrightarrow \Delta}\ \mathsf{refl}$$

This way we obtain sequent calculi for quantified **D** (characterized by the axiom D), **T** (T), **S4** ($T, 4$), **S4.2** ($T, 4, 2$), **S4.3** ($T, 4, 3$).

Proposition 1 (Main proposition).
Let $\mathcal{TL}_{\mathbf{L}}$ denote $\mathcal{TL}_{\mathbf{D}}$, $\mathcal{TL}_{\mathbf{T}}$, $\mathcal{TL}_{\mathbf{S4}}$, $\mathcal{TL}_{\mathbf{S4.2}}$ or $\mathcal{TL}_{\mathbf{S4.3}}$. Then

1. *$\mathcal{TL}_{\mathbf{K}}$ is sound and complete for quantified **K** on constant domains with rigid designators;*
2. *$\mathcal{TL}_{\mathbf{L}}$ is sound for quantified **L** on constant domains with rigid designators.*

(this result appears in a forthcoming paper, currently under revision by the *Journal of Logic and Computation*.)

The remarkable property of the family of sequent calculi $\mathcal{TL}_{\mathbf{L}}$ is that they are modular in the Kripke frame: to obtain the calculus for a stronger logic, we just need to add one or more sequent rules to the previous calculus. We are currently working on completeness for item #2. This work owes a lot to Basin, Matthews and Viganò's early works on labelled deduction in modal logics (see [BMV97a,BMV97b,BMV98]).

So far for modal logics; but, as we are interested in temporal logics, mainly with a frame isomorphic to the natural numbers and with a "next" operator, we consider a further extension. We introduce a unary function σ, called *successor*, in the labelling language, and let the label interpretation I_l take care of its semantics. To use this function in the new calculus, which we will call $\mathcal{TL}_{\mathsf{ind}}$, a new modal operator $\bigcirc$ is introduced, together with appropriate rules:

$$\frac{\Gamma, \varphi@\sigma(\tau) \longrightarrow \Delta}{\Gamma, \bigcirc\varphi@\tau \longrightarrow \Delta}\ l\bigcirc \qquad \frac{\Gamma \longrightarrow \varphi@\sigma(\tau), \Delta}{\Gamma \longrightarrow \bigcirc\varphi@\tau, \Delta}\ r\bigcirc$$

We also add the cut rule, rules for induction and rules which define the relation between $\preceq$ and σ. Some examples:

$$\frac{\Gamma \longrightarrow \varphi@\tau, \Delta \quad \Gamma, \tau \preceq t_a, \varphi@t_a \longrightarrow \varphi@\sigma(t_a), \Delta}{\Gamma \longrightarrow \Box\varphi@\tau, \Delta}\ r\mathsf{ind}$$

$$\frac{\Gamma \longrightarrow \tau_2 \preceq \tau_1, \Delta}{\Gamma, \tau_1 \preceq \tau_0 \longrightarrow \tau_2 \preceq \tau_0, \Delta}\ \sigma_1$$

The optimal set of rules is the object of our future research. We suspect that a set of rules enforcing a full Presburger Arithmetic decision procedure would make $\mathcal{TL}_{\mathsf{ind}}$ equivalent to Annotated Temporal Logic [MMW94], which is complete for a wide subclass of linear time first-order temporal logic, defined in [AM90]. We also suspect that $\mathcal{TL}_{\mathsf{ind}}$ embraces a logic which is stronger than **S4.3.1**, obtained by adding the Dummett axiom to **S4.3** (see [HC96] and [Gor93] for a thorough exposition of the hierarchy of modal and temporal logics and their axiomatizations).

3 System description and experiments

FTL is a prototypal implementation of $\mathcal{TL}_{\mathbf{L}}$ in in λProlog. λProlog is a higher-order logic programming language which allows logical modules, λ-abstraction and -application and higher-order hereditary Harrop formulae in place of first-order Horn clauses as in ordinary Prolog [NM98]. λProlog fits well the needs of automated theorem proving; our implementation is largely based on Amy Felty's work on classical and intuitionistic theorem proving [Fel93]. In that paper a proof of correctness of the implementation is given, and it is easy to adapt such a proof for our case. The proof employs a translation between the object logic and the higher-order logic of λProlog, showing that every proof in $\mathcal{TL}_{\mathbf{L}}$ corresponds to a higher-order term in the meta-logic and viceversa.

FTL consists of five λProlog modules. Modules dynamically make their clauses available to other modules, and this mechanism is particularly well suited in this case, where we want the machinery for reasoning on time to be as opaque as possible with respect to the prover.

The sorts of the object logic are defined in λProlog straightforwardly; Boolean connectives are logical formulae (`sformula`) constructors:

```
type neg      sformula -> sformula.
type imp      sformula -> sformula -> sformula.
type glob     sformula -> sformula.
type forall   (i -> sformula) -> sformula.
```

As one can see in the definition of $\forall$ (`forall`), λ-abstractions provide a representation of object-level quantification: a logical formula $\forall x.p(x)$ becomes `forall x\ (p x)` (the backslash encodes λ-abstraction in λProlog). In this example, `p` has been previously declared as a unary predicate (`type p  i -> sformula.`).

The quantified variable is abstracted away, causing λProlog's unification algorithm, which coincides with $\beta\eta$-reduction, to detect equivalence among formulae regardless of bound variables.

Tactic theorem proving means that we want to establish a relation between a sequent and its proof (namely, that the proof proves the sequent). We try to reach this goal by means of steps enforced by tactics. We thus define a *sequent* as a pair of lists of formulae, and a *goal* as a pair sequent / proof:

```
kind  goal      type.
kind  proof     type.
kind  sequent   type.

type  -->       (list formula) -> (list formula) -> sequent.
type  proves    proof -> sequent -> goal.
```

Among other apt features, λProlog's meta-level universal quantification can be used to enforce the provisos on universal force rules ($r\forall$, $r\Box$, $r$ind and so on). Whenever this mechanism is used, a fresh term is introduced, which is forbidden to unify with any other term already present in the program, including logical variables.

A *tactic* is a predicate linking two goals (its type being therefore `goal -> goal -> o`), stating that one of them can be reached from the other by means of some (multiple) rule application. Besides having imported from Felty's work *compound* tactics, which enforce repeated, exhaustive and conditional application of tactics, we have written roughly one basic tactic per sequent rule, such as in this example (the embedded rule is $r\supset$):

```
type r_imp          proof -> proof.
type r_imp_tac      goal -> goal -> o.

r_imp_tac ((r_imp P) proves (Gamma --> Delta))
          (P proves (((Phi @ Tau)::Gamma) --> ((Psi @ Tau)::Delta'))) :-
  delete (Phi imp Psi @ Tau) Delta Delta'.
```

An intuitive reading of the tactic goes as follows: if the conclusion set Δ contains $\varphi \supset \psi@\tau$ then we remove it and call the new conclusion set Δ'. If this operation was successful, proof `(r_imp P)` proves sequent $\Gamma \longrightarrow \Delta$, provided that proof `P` proves sequent $\Gamma, \varphi@\tau \longrightarrow \psi@\tau, \Delta'$.

Object-level proofs become meta-level *terms*, and this correspondence is the basis of a proof of correctness of our implementation (directly adapted from the one given in [Fel93]): it is shown that every object-level proof corresponds to a higher-order term in the meta-logic and viceversa, thus establishing that a sequent S is provable if and only if there is a corresponding λProlog term `P` such that the relation `P proves` $||S||$ holds, where $||\cdot||$ denotes the encoding of the sequent as a term of the meta-logic.

Reasoning on the Kripke frame (on the structure of time) is almost totally confined in one of the five modules, in the spirit of the sequent calculi themselves. The tactic for rule $r$ind, for instance, goes as follows:

```
type r_ind       proof -> (time -> proof) -> proof.
type r_ind_tac   goal -> goal -> o.

r_ind_tac ((r_ind P1 P2) proves (Gamma --> Delta))
          (and_goal
            (P1 proves (Gamma --> ((Phi @ Tau)::Delta')))
            (forall_goal ta\
              ((P2 ta) proves ((Tau wbefore ta)::(Phi @ ta)::Gamma) -->
                           ((Phi @ (s ta))::Delta')))) :-
  delete (glob Phi @ Tau) Delta Delta'.
```

where `wbefore` denotes the accessibility relation $\preceq$ and `s` is the successor function. The tactic mimicks the standard basic induction principle: to prove $\Box\varphi@\tau$ we prove that $\varphi@\tau$ holds, and that if $\varphi@t_a$ and $\tau \preceq t_a$ hold for a fresh time term t_a, $\varphi@s(t_a)$ must hold, too.

3.1 Experiments

FTL is still at an early stage of development. Nonetheless it has been able to prove automatically all relevant valid formulae taken from [AM90] and [MP81], plus the simple specification of a Boolean circuit, the proof of the Dummett axiom and of a non-trivial inductive statement called *simplified whisky problem*[1]. The proof of the Dummett axiom (not completely automatic), in particular, lets us suspect that $\mathcal{TL}_{\mathsf{ind}}$ could be complete for a large subset of **S4.3.1**.

As an example, we give below the definition and sketch of the proof of the simplified whisky problem. Let p be a unary predicate, a a constant and f a unary function; we want to prove the validity of

$$p(a) \ \wedge \ \Box\forall x.[p(x) \supset p(f(x))] \ \wedge \ \Box\forall x.[p(f(x)) \supset \bigcirc p(x)] \quad \supset \quad \Box p(a)$$

The intuition of this formula is: $p(a)$ holds at time zero, for all x it is always the case that $p(x)$ implies $p(f(x))$, and for all x it is always the case that $p(f(x))$ implies $p(x)$ at the next instant. The first conjunct can be seen as the initial condition, whereas the two following conjuncts are *invariants* (they always hold). It is now clear that the initial condition, plus the two invariants, imply the conclusion "$p(a)$ is always true". Ideally a proof of this formula would require a kind of "double" induction, on time and on the function f. Figure 2 illustrates the situation.

We define a simple exhaustive compound tactic which applies rules $r$ind and $l$ind after closing (ax and ent) and non-branching rules (such as $l\wedge$):

```
auto_tac InGoal OutGoal :-
  exhaust_tac ( <... closing tactics ...> ::
                <... non-branching tactics ...> ::
                <... induction tactics ...> ::
                <... other tactics ...> :: nil)
  InGoal OutGoal.
```

[1] The name of this problem is due to Regimantas Pliuskevicius.

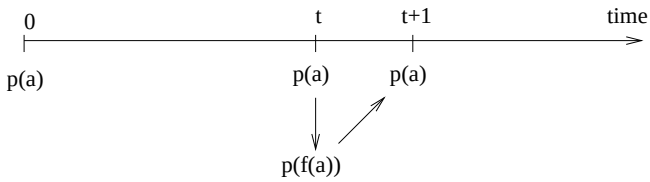

Fig. 2. an intuitive reading of the simplified whisky problem. Induction on f is "chained" to induction on time.

where compund tactic `exhaust_tac` repeatedly and exhaustively applies a list of tactics to a formula. FTL soon obtains the "flattened" sequent

$$p(a)@0, \Box\forall x.[p(x) \supset p(f(x))]@0, \Box\forall x.[p(f(x)) \supset \bigcirc p(x)]@0 \longrightarrow \Box p(a)@0$$

At this point, the induction rule opens two branches. The base case, in which $p(a)@0$ appears among the conclusions, is trivial, while the step case requires 14 basic tactics and generates 5 branches. The resulting proof is, remarkably, very intuitive — prettyprinted in a tree-like shape, it can be followed by a human reader.

4 Conclusions

The original contribution of our work are so far represented by the sequent calculi up to quantified **S4.3** and **S4.3.1**, which had never been reached so far to our knowledge, and the implementation, which constitutes a novel application of tactical theorem proving in a higher-order setting to labelled deductive systems and first-order modal and temporal logics in particular.

Future work primarily includes the application of the proof planning paradigm to FTL via integration with the λCLAM system [RSG98]. We aim to use proof planning to guide the search in these calculi, and in particular to help with the difficult problem of controlling the interaction of the cut rule with our induction rules for first-order temporal logic.

An interesting issue is raised by the "whisky problem", which is a weaker version of the problem stated in the previous Section:

$$p(a) \ \wedge \ \forall x.[p(x) \supset p(f(x))] \ \wedge \ \Box\forall x.[p(f(x)) \supset \bigcirc p(x)] \quad \supset \quad \Box p(a).$$

Here the second conjunct is not an invariant, and nonetheless the formula is still valid, as the intuition given in Figure 3 explains. It seems that some kind of higher-order abstraction over proofs is necessary to prove this formula; we think λCLAM could help.

Finally, our long-term aim is automated system verification, so we also plan to investigate the use of tactics to find loop invariants and aid requirement strengthening. At that point, we plan to run comparative tests with such systems as STeP [M$^+$95].

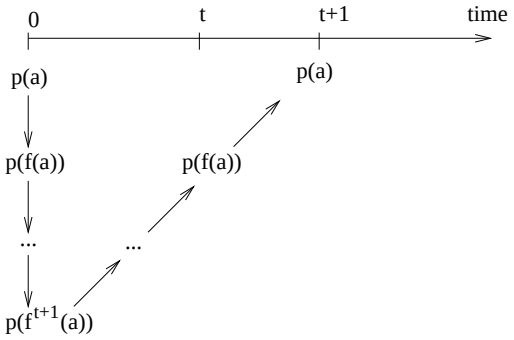

Fig. 3. an intuitive reading of the whisky problem. Some kind of higher-order abstraction over proofs is needed to handle objects such as $p(f^{t+1}(a))$.

Acknowledgments

This work is being carried out at the University of Edinburgh and is supported by the EPSRC Grant GR/M46624, "Mechanising First-Order Temporal Logics". The authors wish to thank all other people involved in the project: Alan Bundy, Anatoli Degtyarev, Paul Jackson, Michael Fisher and Peter Quigley.

References

[AM90] Martin Abadí and Zohar Manna. Nonclausal deduction in first-order temporal logic. *Journal of the ACM*, 37(2):279–317, April 1990.

[BD92] Cristina Bicchieri and Maria Luisa Dalla Chiara, editors. *Knowledge, belief, and Strategic Interaction*. Cambridge University Press, Cambridge, England, 1992.

[BMV97a] D. Basin, S. Matthews, and L. Vigano. A new method for bounding the complexity of modal logics. *Lecture Notes in Computer Science*, 1289:89–??, 1997.

[BMV97b] David Basin, Seán Matthews, and Luca Viganò. Labelled propositional modal logics: Theory and practice. *Journal of Logic and Computation*, 7(6):685–717, December 1997.

[BMV98] David Basin, Seán Matthews, and Luca Viganò. Labelled modal logics: Quantifiers. *Journal of Logic, Language, and Information*, 7(3):237–263, 1998.

[BO95] F. Baader and H.-J. Ohlbach. A multi-dimensional terminological knowledge representation language. *J. Applied Non-Classical Logics*, 5:153–197, 1995.

[Fel93] Amy Felty. Implementing tactics and tacticals in a higher-order logic programming language. *Journal of Automated Reasoning*, 11(1):43–81, August 1993.

[Gab96] Dov M. Gabbay. *Labelled Deductive Systems, Volume 1.* Oxford University Press, Oxford, 1996.

[GHR93] Dov M. Gabbay, Ian Hodkinson, and Mark Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects. Volume 1: Mathematical Foundations.* Oxford University Press, *forthcoming*, 1993.

Preprinted as Tech Report MPI-I-92-213, Max-Planck-Institut für Informatik, Saarbrücken, Germany.

[Gor93] R. Gore. Cut-free sequent and tableau systems for propositional diodorean modal logics. Technical Report UMCS-93-8-3, University of Manchester, Computer Science Department, August 1993.

[HC96] G. E. Hughes and M. J. Cresswell. *A New Introduction to Modal Logic*. Routledge, London and New York, 1996.

[M+95] Z. Manna et al. STeP: The Stanford Temporal Prover. In Peter D. Mosses, Mogens Nielsen, and Michael I. Schwartzbach, editors, *TAPSOFT '95 : theory and practice of software development*, volume 915 of *Lecture Notes in Computer Science*, pages 793–794. Springer, 1995.

[MMW94] H. McGuire, Z. Manna, and B. Waldinger. Annotation-based deduction in temporal logic. In Dov M. Gabbay and Hans Jürgen Ohlbach, editors, *Proceedings of the 1st International Conference on Temporal Logic*, volume 827 of *LNAI*, pages 430–444, Berlin, July 1994. Springer.

[MP81] Zohar Manna and Amir Pnueli. Temporal verification of concurrent programs: the temporal framework for concurrent programs. In R.S̃. Boyer and J Strother Moore, editors, *The Correctness Problem in Computer Science*, chapter 5, pages 215–273. Academic Press, London, 1981.

[MP92] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer, New York, 1992.

[NM98] Gopalan Nadathur and Dale Miller. Higher-order logic programming. In Dov M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logics for Artificial Intelligence and Logic Programming*, volume 5, pages 499–590. Clarendon Press, Oxford, England, 1998.

[RSG98] Julian Richardson, Alan Smaill, and Ian Green. Proof planning in higher-order logic with lambda-clam. In Claude Kirchner and Hélène Kirchner, editors, *Proceedings of the 15th International Conference on Automated Deduction (CADE-98)*, volume 1421 of *LNAI*, pages 129–133, Berlin, July 5–10 1998. Springer.

[van84] Johan van Benthem. Correspondence theory. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, Volume II: Extensions of Classical Logic*, volume 165 of *Synthese Library*, chapter II.4, pages 167–247. D. Reidel Publishing Co., Dordrecht, 1984.

[Wal89] L. A. Wallen. *Automated Deduction in Non-Classical Logics*. MIT Press, 1989.