

**Ragionamento Automatico per
Logiche Modali e Temporal Quantificate**
*Automated Reasoning in
Quantified Modal and Temporal Logics*

Claudio Castellini

SOMMARIO/ABSTRACT

Descriviamo in questo articolo la tesi di dottorato dell'autore, centrata sul ragionamento automatico nelle logiche modali e temporali quantificate. I contributi originali della tesi sono: (i) la formulazione di una famiglia di calcoli di sequenti corretti e completi per le logiche modali quantificate; (ii) l'estensione dell'approccio alla logica temporale quantificata del tempo lineare e discreto, e la creazione di un framework per il ragionamento automatico in essa basato sul Proof Planning; (iii) risultati sperimentali ottenuti applicando il suddetto framework al problema delle Feature Interactions nei sistemi di telecomunicazioni.

We report on the author's Ph.D. thesis, concerned with automated reasoning in quantified modal and temporal logics. The relevant contributions are three: (i) we devise a sound and complete set of sequent calculi for quantified modal logics; (ii) we extend the approach to the quantified temporal logic of linear, discrete time and develop a framework for doing automated reasoning via Proof Planning in it; (iii) we show a set of experimental results obtained by applying the framework to the problem of Feature Interactions in telecommunication systems.

Keywords: automated reasoning, quantified modal logics, quantified temporal logics, proof planning, feature interactions

1 Introduction

Modal and temporal logics are extensions of classical logic in which the notion of *truth* is richer and more subtle. They have been a fascinating philosophical and theoretical subject since their invention by Aristotle. But more recently, especially after Hintikka and Kripke's idea of a "possible worlds" semantics [18, 22], they have been also applied, especially in formal verification. Temporal logics in particular can capture the behaviour of a number of complex

systems (circuits, protocols, programs) the safety of which has to be verified.

However, while propositional modal and temporal logics have been widely studied, *quantified* modal and temporal logics have been quite neglected since quantification introduces a lot of complexity. As an example, consider that the propositional temporal logic of linear, discrete time, *LTL*, is well known to be decidable, whereas its quantified counterpart, that we will call **FOLTL**, is not only undecidable, but non-recursively enumerable [19].

Nevertheless, the expressivity added by quantification makes automated reasoning in these logics an appealing instrument for formal verification. Consider for instance **FOLTL**: such a logic offers the possibility of expressing in a natural and compact way the temporal behaviour of such complex systems as security protocols, multi-agent environments, infinite-state programs and so on. Complex data types can be built out of recursive definitions, and fairness constraints can be easily imposed on the model, thanks to the expressivity of the language.

In this paper we report on the author's Ph.D. thesis, concerned exactly with quantified modal and temporal logics, and automated reasoning in them. The paper's structure follows the three main contribution of the thesis, that is:

- Section 2 presents a family of Gentzen-style sequent calculi for quantified modal logics (QMLs) which enjoy some theoretically and practically relevant properties. *Any* QML whose Kripke frame can be axiomatised in first-order logic with equality is captured by the framework.
- Section 3 describes the architecture of an automated reasoning system for **FOLTL** based upon proof planning [2]. The system is centered around a simple, interactive theorem prover for **FOLTL** and the proof planner λCIAM [27], which drives it.
- Section 4 shows the case-study chosen in order to demonstrate the effectiveness of the proposed ap-

proach. The problem of *Feature Interactions* in telecommunications systems [17] is modelled in **FOLTL** via an intuitive and clear set of formulae, and it is shown that the proof planning approach solves a set of associated formal verification problems, without making any simplifying assumption of finiteness over the domain.

Lastly, Section 5 draws conclusions and outlines related and future work.

2 Sequent Calculi for QMLs

In the following, we will be considering *normal* QMLs on constant domains. Consider any such logic and the associated class of Kripke frames which characterise it: it is well known from Correspondence Theory [30] that a subset of these logics are characterised by frames whose properties can be axiomatised in first-order logic with equality, where the language contains objects to denote the possible worlds and a binary functional symbol, R , denoting the accessibility relation. Call this subset of logics *FO-axiomatisable* QMLs. Remarkable examples of FO-axiomatisable QMLs are:

- **QT**, the quantified logic of reflexive frames, axiomatised by the first-order sentence $\forall x.xRx$ (reflexivity).
- **QS4.3**, the quantified logic of reflexive, transitive, strongly connected frames, i.e., linear partial orders, axiomatised by the sentence above plus $\forall t_0 t_1 t_2.(t_0 \prec t_1 \wedge t_1 \prec t_2) \supset t_0 \prec t_2$ (transitivity) and $\forall t_1 t_2.t_1 \prec t_2 \vee t_1 = t_2 \vee t_2 \prec t_1$ (strong connectedness). Notice that the last sentence employs disjunction and equality.
- **QS4.1**, the quantified logic of reflexive, transitive, *atomic* frames, i.e., frames in which every R -chain has a final, reflexive element, axiomatised by the axioms of reflexivity and transitivity (see above) and $\forall t_1 \exists t'.t_1 \prec t' \wedge \forall t_2.t' \prec t_2 \supset t' = t_2$ (atomicity).

(Of course, nothing prevents the logician from creating new, ad-hoc such logics, by defining a set of first-order properties needed by the frames which should characterise the logic.) Since such a rich set of QMLs can be characterised via simple sets of first-order sentences (i.e., formulae with no free variables), we investigate the possibility of building a sequent calculus for a FO-axiomatisable logic by turning the sentences defining the properties of its frame into a set of new sequent rules.

It turns out that for *any* FO-axiomatisable whatsoever, a (labelled) sequent calculus can be easily built, which is *sound and complete* for the logic, and which additionally enjoys some properties which are useful for the mechanisation of the logic.

Let us go more in detail. Assume the standard notions of logical language for modal and first-order logic; assume

also that we have a separate, first-order language to denote worlds of the Kripke frame, equality among them, $=$, and the accessibility relation, R ; the notation $\varphi @ \tau$ will indicate that the quantified modal formula φ holds at the world denoted by the label τ (more about labelled deduction can be found in the work of Dov Gabbay, e.g. [13]). Then Table 1 shows $\mathcal{C}_{\mathbf{QK}}$, a sound and complete sequent calculus for the weakest quantified normal modal logic **QK**, characterised by the class of all frames.

$\mathcal{C}_{\mathbf{QK}}$ is a variant of Gentzen’s sequent calculus LK for classical logic [15], except that formulae are labelled, and that there are two rules $r\Box$ and $l\Box$ for the modal operator \Box , intuitively reflecting its semantics.

Consider for instance rule $r\Box$, read bottom-up: to prove that $\Box\varphi @ \tau$ (“ φ holds at all worlds accessible from τ ”), one assumes that $\tau \prec t'$ (“ t' is accessible from τ ”) and reduces to the goal of proving $\varphi @ t'$, where t' is fresh, denoting any generic world. This calculus is well known and, with small variations, already appears in the literature at least some ten years ago, e.g., in [1] and [28].

In order to take equality into account, we also need rules in Table 2; call $\mathcal{C}_{\mathbf{QK}}^=$ the calculus resulting from the union of the rules of $\mathcal{C}_{\mathbf{QK}}$ and the rules visible in the Table. For more details, see [11]; equality in sequent calculi for first-order logic has been first dealt with, as far as we know, by Kanger [21].

Table 2: rules for equality. $\mathcal{C}_{\mathbf{QK}}^=$ is the union of these rules and $\mathcal{C}_{\mathbf{QK}}$.

$$\frac{}{\Gamma \longrightarrow \tau = \tau, \Delta} \text{ refl}_=$$

$$\frac{\Gamma[\tau'/t], \tau = \tau' \longrightarrow \Delta[\tau'/t]}{\Gamma[\tau/t], \tau = \tau' \longrightarrow \Delta[\tau/t]} \text{ sub}_=$$

2.1 The Strengthening Procedure

Now the question is: how can we extend this calculus toward stronger QMLs, that is, with a more structured frame¹? We first introduce a procedure which turns a first-order sentence in a sequent rule. Let $2LK$ be the standard extension of Gentzen’s LK to two-sorted first-order logic with equality, analogously, for instance, to the calculus $G_=_$ presented in [14], Definition 10.5.1. Then

Definition 1 (Strengthening) *Let ϕ be a first-order sentence; then the strengthening procedure, yielding sequent rule $\text{Str}(\phi)$, is defined as follows:*

¹the word *stronger* simply means “larger” as far as a logic is considered as the smallest set of formulae which are true under the associated semantics. For instance, **QT** is stronger than **QK** in that the modal axiom of reflexivity, $\Box p \supset p$, is true in the former but not in the latter.

Table 1: the calculus $\mathcal{C}_{\mathbf{QK}}$ for \mathbf{QK} . a, t' cannot appear free in the conclusion of $r\forall$ and $r\Box$.

$$\begin{array}{c}
 \overline{\Gamma, A \longrightarrow A, \Delta} \text{ ax} \\
 \\
 \frac{\Gamma \longrightarrow \varphi @ \tau, \Delta}{\Gamma, \neg\varphi @ \tau \longrightarrow \Delta} l\neg \qquad \frac{\Gamma, \varphi @ \tau \longrightarrow \Delta}{\Gamma \longrightarrow \neg\varphi @ \tau, \Delta} r\neg \\
 \frac{\Gamma, \psi @ \tau \longrightarrow \Delta \quad \Gamma \longrightarrow \varphi @ \tau, \Delta}{\Gamma, \varphi \supset \psi @ \tau \longrightarrow \Delta} l\supset \qquad \frac{\Gamma, \varphi @ \tau \longrightarrow \psi @ \tau, \Delta}{\Gamma \longrightarrow \varphi \supset \psi @ \tau, \Delta} r\supset \\
 \frac{\Gamma, \forall x. \varphi @ \tau, \varphi[c/x] @ \tau \longrightarrow \Delta}{\Gamma, \forall x. \varphi @ \tau \longrightarrow \Delta} l\forall \qquad \frac{\Gamma \longrightarrow \varphi[a/x] @ \tau, \Delta}{\Gamma \longrightarrow \forall x. \varphi @ \tau, \Delta} r\forall \\
 \frac{\Gamma, \Box\varphi @ \tau, \varphi @ \tau' \longrightarrow \Delta \quad \Gamma, \Box\varphi @ \tau \longrightarrow \tau < \tau', \Delta}{\Gamma, \Box\varphi @ \tau \longrightarrow \Delta} l\Box \qquad \frac{\Gamma, \tau < t' \longrightarrow \varphi @ t', \Delta}{\Gamma \longrightarrow \Box\varphi @ \tau, \Delta} r\Box
 \end{array}$$

1. convert ϕ into prenex normal form and Skolemise; call the new sentence ϕ^S ;
2. build a 2LK-derivation of $\Gamma, \phi^S \longrightarrow \Delta$ in which every sequent labelling a leaf contains only constraints, Γ or Δ (when eliminating the universal quantifier on the left-hand side, avoid copying the main formula into the premise);
3. finally, let $\Gamma \longrightarrow \Delta$ be the conclusion of $\text{Str}(\phi)$, and let the leaves of the derivation obtained at the previous step be its premises.

The idea is that any first-order sentence can be turned into a sequent rule just by “unfolding” it and taking the root and leaves of the resulting derivation. As an example, consider axiom 2, defining *weak directedness*:

$$\forall t_0 t_1 t_2. (t_0 < t_1 \wedge t_0 < t_2) \supset \exists t'. t_1 < t' \wedge t_2 < t'$$

Figure 1 shows how to unfold it. Rule $\text{Str}(2)$ is built by taking $\Gamma \longrightarrow \Delta$ as the conclusion, and the leaves of the proof tree in Figure 1 as the premises (notice the introduction of the Skolem function cv). The strengthening procedure is, almost trivially, terminating (see [10] for a proof).

Now let \mathbf{QL} be a QML stronger than \mathbf{QK} , whose frame properties are axiomatised by the set of first-order sentences (with equality) $\text{FrmAx}(\mathbf{QL})$; then let $\text{Str}(\mathbf{QL})$ be the set of sequent rules $\text{Str}(\mathbf{QL}) = \{\rho \mid \rho = \text{Str}(\phi), \phi \in \text{FrmAx}(\mathbf{QL})\}$. A sequent calculus for \mathbf{QL} is then built by taking $\mathcal{C}_{\mathbf{QK}}^{\overline{}} \cup \text{Str}(\mathbf{QL})$.

This calculus is obviously *modular*, in that it is obtained by adding to the (unchanged) basic calculus $\mathcal{C}_{\mathbf{QK}}$ a set of new rules, and *uniform*, in that (as Definition 1 suggests) each sequent rule in $\text{Str}(\mathbf{QL})$ is clearly and intuitively related to a first-order sentence enforcing a property of the frame. Moreover, since the strengthening procedure

is terminating, and since $\text{FrmAx}(\mathbf{QL})$ is finite, we have that calculi obtained this way are *finitary* — they have a finite number of rules, each rule has a finite number of premises, and each sequent is finite. Besides adding to the elegance of the presentation, modularity and uniformity are useful for the implementation of these logics — such an implementation would indeed benefit from not having to be redone from scratch each time a new, stronger logic is needed; modularity of the calculi can be reflected in modularity of the automated machinery.

2.2 Normalisation: the entailment rule

One more step is worth being taken: we introduce a rule which “groups” all frame rules. Let $\text{FrmRI}(\mathbf{QL})$ be the union of $\text{Str}(\mathbf{QL})$ and the rules in Table 2; then

Definition 2 ($\mathcal{C}_{\mathbf{QL}}$ and the entailment rule) *Let \mathbf{QL} be any FO-axiomatisable logic; let then*

$$\mathcal{C}_{\mathbf{QL}} = \mathcal{C}_{\mathbf{QK}} \cup \text{ent}_{\mathbf{QL}}$$

where $\text{ent}_{\mathbf{QL}}$ (entailment) is the following rule:

$$\overline{\Gamma \longrightarrow \Delta} \text{ ent}_{\mathbf{QL}} \quad \text{with } \vdash_{\text{FrmRI}(\mathbf{QL})} \Gamma \longrightarrow \Delta.$$

According to the above Definition, in each $\mathcal{C}_{\mathbf{QL}}$ -proof, rule $\text{ent}_{\mathbf{QL}}$ represents a subproof in which only rules in $\text{FrmRI}(\mathbf{QL})$ are used. The calculi $\mathcal{C}_{\mathbf{QL}}$, with respect to the calculi $\mathcal{C}_{\mathbf{QK}}^{\overline{}} \cup \text{Str}(\mathbf{QL})$, have a restriction on the use of frame rules; since $\text{ent}_{\mathbf{QL}}$ is a closing rule, it cannot be followed higher up in the tree by the application of any logical rule. In other words,

Proposition 3 (Normalisation) *Let \mathbf{QL} be any FO-axiomatisable logic; then every $\mathcal{C}_{\mathbf{QL}}$ -proof is normal, in the sense that no logical rules are ever used above a frame rule.*

$$\begin{array}{c}
\frac{\Gamma \longrightarrow \tau_0 \prec \tau_1, \Delta \quad \Gamma \longrightarrow \tau_0 \prec \tau_2, \Delta}{\Gamma, \longrightarrow \tau_0 \prec \tau_1 \wedge \tau_0 \prec \tau_2, \Delta} \quad r\wedge^* \quad \frac{\Gamma, \tau_1 \prec cv(\tau_0, \tau_1, \tau_2), \tau_2 \prec cv(\tau_0, \tau_1, \tau_2) \longrightarrow \Delta}{\Gamma, \tau_1 \prec cv(\tau_0, \tau_1, \tau_2) \wedge \tau_2 \prec cv(\tau_0, \tau_1, \tau_2) \longrightarrow \Delta} \quad l\wedge^* \\
\frac{\Gamma, \tau_1 \prec cv(\tau_0, \tau_1, \tau_2) \wedge \tau_2 \prec cv(\tau_0, \tau_1, \tau_2) \longrightarrow \Delta}{\Gamma, (\tau_0 \prec \tau_1 \wedge \tau_0 \prec \tau_2) \supset \tau_1 \prec cv(\tau_0, \tau_1, \tau_2) \wedge \tau_2 \prec cv(\tau_0, \tau_1, \tau_2) \longrightarrow \Delta} \quad l\supset^* \\
\frac{\Gamma, (\tau_0 \prec \tau_1 \wedge \tau_0 \prec \tau_2) \supset \tau_1 \prec cv(\tau_0, \tau_1, \tau_2) \wedge \tau_2 \prec cv(\tau_0, \tau_1, \tau_2) \longrightarrow \Delta}{\Gamma, \forall t_0 t_1 t_2. (t_0 \prec t_1 \wedge t_0 \prec t_2) \supset t_1 \prec cv(t_0, t_1, t_2) \wedge t_2 \prec cv(t_0, t_1, t_2) \longrightarrow \Delta} \quad l\forall_\theta^* l\forall_\theta^* l\forall_\theta^*
\end{array}$$

Figure 1: application of Step 2 of the strengthening procedure to the sentence 2^S .

All calculi $\mathcal{C}_{\mathbf{QL}}$ obviously retain the properties describe above (modularity, uniformity and finitariness). Moreover, the property of normalisation reduces the search space during proof search in any $\mathcal{C}_{\mathbf{QL}}$. In principle, rule $\text{ent}_{\mathbf{QL}}$ can be replaced by any reasoning method whatsoever for the first-order theory of $\text{FrmAx}(\mathbf{QL})$, seen as a black box; in particular, any efficient machinery for equivalence reasoning can be employed.

2.3 Soundness and Completeness

Quite surprisingly, it turns out that $\mathcal{C}_{\mathbf{QL}}$ is sound and complete for \mathbf{QL} , for any $\mathcal{C}_{\mathbf{QL}}$. The core of the proof is a simple translation $\llbracket \cdot \rrbracket$ of the formulae of \mathbf{QL} into two-sorted first-order logic with equality, call it $2FOL$ (see [10] for the full details). In particular, rather than proving the above equivalence directly, we show that these four items are equivalent: (i) $\Gamma \longrightarrow \Delta$ is a theorem of $\mathcal{C}_{\mathbf{QL}}$, (ii) $\llbracket \Gamma \cup \text{FrmAx}^S(\mathbf{QL}) \longrightarrow \Delta \rrbracket$ is a theorem of $2LK$, (iii) $\llbracket \Gamma \cup \text{FrmAx}(\mathbf{QL}) \longrightarrow \Delta \rrbracket$ is valid in $2FOL$, and (iv) $\Gamma \longrightarrow \Delta$ is valid in \mathbf{QL} , where $\text{FrmAx}^S(\mathbf{QL})$ denotes the Skolemised version of $\text{FrmAx}(\mathbf{QL})$, the first-order sentences axiomatising the logic \mathbf{QL} .

(i) if and only if (ii). Let two proofs be called *similar* if they prove the same sequent, and let $2LK$ be a simple extension of Gentzen’s sequent calculus LK , adapted to $2FOL$. Then it can be shown that every $\mathcal{C}_{\mathbf{QL}}$ -proof is similar to a $2LK$ -proof of a $2FOL$ -translated sequent, and vice-versa. This proves the equivalence, since it means that all which is provable in $\mathcal{C}_{\mathbf{QL}}$ is provable modulo $\llbracket \cdot \rrbracket$ in $2LK$; and all which is the $\llbracket \cdot \rrbracket$ -translation of something and is provable in $2LK$, is provable modulo $\llbracket \cdot \rrbracket$ in $\mathcal{C}_{\mathbf{QL}}$.

(ii) if and only if (iii). $2LK$ is sound and complete for $2FOL$ (Lemma 10.5.1, Theorem 10.5.1 in [14]; Section 1 of [11]). This proves the equivalence, since the $2FOL$ theory of $\text{FrmAx}^S(\mathbf{QL})$ is a conservative extension of that of $\text{FrmAx}(\mathbf{QL})$ (see, e.g., [29], p. 55).

(iii) if and only if (iv). It suffices to prove the equivalence for single formulae. The equivalence is proved by showing that, given a model in \mathbf{QL} for a formula φ , there is a corresponding model for $\llbracket \varphi \rrbracket$ in $2FOL$, and vice-versa.

3 Automated Reasoning in FOLTL

The paradigm of Proof Planning revolves around the idea of first finding the “outline” of a proof (a *proof plan*) which

has then to be refined into a proof. Search is lifted to an abstract space, smaller than the concrete one and in which little or no backtracking is likely to occur. The process can fail both at the planning level (no plan found) or at the proving level (the plan could not be refined to a proof); therefore, one wants to use it for very complex logics, as is the case of **FOLTL**.

Proof planning needs (i) a proof system for the object logic being considered; (ii) a theorem prover for it, with little or no automation, but with a lot of control capabilities; (iii) a proof planner and a set of methods enforcing high-level steps of reasoning for the problem considered, and a tight coupling between the planner and the prover. We have first built a (labelled) sequent calculus for **FOLTL** following the ideas seen in the previous Section, obviously giving up completeness since the logic is non recursively enumerable [19] and our calculi are finitary. Then we have written a simple, interactive theorem prover for **FOLTL**, called **FTL**. The prover is written in the higher-order logic programming language λProlog [23, 25, 24] and enjoys a number of features such as higher-order unification, modularity (which reflects the modularity of our approach with sequent calculi) and λ -abstraction and application. Lastly, the proof planner λCIAM [27], has been adapted to drive **FTL**, that is, to use it in order to refine its proof plans to **FOLTL** proofs. The sequence of operations goes as follows:

1. the input sequent is written in the input language of **FTL**, a straight coding of **FOLTL**;
2. the input sequent is translated to λCIAM ’s internal syntax;
3. λCIAM ’s proof plan engine is called, and hopefully a proof plan for the translation of the input sequent is returned;
4. the proof plan is translated to a tactic and passed to **FTL**, which finally
5. applies the tactic to the input sequent and checks that the resulting proof does prove it.

There is only one item which deserves more explanations, and it is Item 4. In Proof Planning, every method has an associated *tactic* which defines the steps the object-level theorem prover must take in order to validate the method itself:

```

method <method-name>
  <input-goal>
  <preconditions>
  <effects>
  <output-goal>
  <tactic>.

```

A method transforms an input goal (that is, a sequent) in an output goal, if the preconditions are met and the effects can be performed. In the simplest case, the operational content of a method refers to the shape of the input goal. For instance: if the input goal starts with a \square and has an existential quantifier in it, then reduce it by removing the operators. Since the method has some knowledge of the shape of the input goal, the associated tactic is instructed in order to apply the sequent rules associated to the operations described above, and then to usually continue via exhaustive propositional reasoning.

In more complex examples, a method can recognise hypotheses which match a certain conclusion and then isolate the relevant formulae; or apply a step of model-checking-like forward search; or even try and apply an induction schema. The delicate pattern recognition issue is devoted to the higher-order unification mechanism enforced in λ Prolog. The algorithm is known to be undecidable [20], but the problem has never arisen in practice.

4 Case-Study and Experimental Results

The case-study we present is an abstraction and simplification of the problem of Feature Interactions in telecommunication systems, a quite well-known hindrance of large telephone networks, which has received quite a lot of attention both from the academic and industrial world (see, e.g., [17, 3, 4]). The phone network is modelled as a set of *users*, each of which enjoys the abilities of answering the phone, dialling a number etc., the so-called *Basic Call Service* (BCS). The environment must take care of establishing connections among users. We model the generic user as a set of **FOLTL** formulae defining the behaviour of the automaton given in Figure 2.

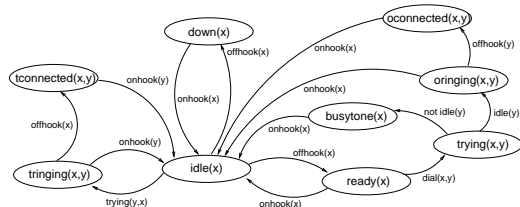


Figure 2: A graphical representation of the BCS.

The expressivity of **FOLTL** keeps the model small and intuitive. The behaviour of the generic user with BCS is enforced via a set of **FOLTL** formulae, among which:

1. (*Initial state*) Every user is initially idle: $\forall x.idle(x)$

2. (*Progress*) For each state, either the user remains in the state forever, or a transition happens; e.g., $\forall x.\square idle(x) \supset idle(x) \mathcal{W} (offhook(x) \vee \exists t.trying(t,x))$
3. (*Trigger*) For each state and transition, if they happen simultaneously then the user will be in a new state at the next instant; e.g., $\forall x.\square idle(x) \wedge offhook(x) \supset (\bigcirc ready(x) \vee \bigcirc down(x))$

This model (*i*) enforces some subtle properties of a real phone network, such as, e.g., that a user that has been called cannot terminate a call, whereas the user who has called can; (*ii*) enjoys a high degree of non-determinism: a state can have more than one successor state even if the action is the same, and as well a user can permanently remain in the current state; (*iii*) there is no restriction on the number of users.

The problem consists of proving whether the above model enjoys a set of desirable properties, or not. As an example, let us concentrate on the simplest properties we considered, i.e., *reachability* properties. We want to check whether, under suitable conditions, a certain state is reachable at all. This corresponds to asking whether the system does enforce its basic requirements, at least in the best case. For example, it must be eventually possible to connect any two users, if the originator dials, if the line is available, if the terminator hangs up and so on.

These properties correspond to looking for a path in the graph of Figure 2 from the initial state to the required state. Assume we can somehow collect all good conditions in a formula $\phi(x)$; then we want to prove that:

Reach1 Under suitable conditions, any user can get ready: $\forall x.\phi(x) \supset \diamond ready(x)$

Reach2 Under suitable conditions, any user can connect any other user: $\forall x.\phi(x) \supset \diamond \exists t.ococonnected(x,t)$

Reach3 Under suitable conditions, any user can be connected to any other user: $\forall x.\phi(x) \supset \diamond \exists t.tconnected(t,x)$

More complex examples include progress and deadlock-freedom properties. Sticking to this simple example, a method is built, which mimics *backward-reachability* in model-checking:

```

method exists_path
repeat:
  1 (init) if we are in the initial state, stop
  2 (trig) otherwise find a trigger formula
      leading to the current state; then
      for each associated transition,
  3 (prog) find a progress formula leading to
      the current transition; for each
      associated state, make it current and
      go back to 1.

```

Consider property **Reach1** (and Figure 2). We start from *ready*; since it is not the initial state (step 1), we find all trigger formulae that can lead to it. There is just one: $\forall x.\square idle(x) \wedge offhook(x) \supset (\bigcirc ready(x) \vee$

$\bigcirc \text{down}(x)$). So, in order to get to *ready*, *idle* and *offhook* must have happened in the past (step 2). The only progress formula related to this is $\forall x. \square \text{idle}(x) \supset \text{idle}(x) \mathcal{W} (\text{offhook}(x) \vee \exists t. \text{trying}(t, x))$ (step 3). So we now know that, if the user is *idle*, under suitable conditions, it will get to *ready*. Since *idle* is the initial state, we are done.

Since our approach is not push-button, it seems fair to give an overview of the time spent by the *user* in devising the approach, beside showing CPU times. We adopt Cantu et al.’s three-fold classification of the time required by the user [12]: *User Time*, spent in formalising a problem, *Proof Time*, spent in tuning proof techniques without modifying the tool, and *Tool Time*, used for debugging. A number of properties have been verified by the system $\lambda\text{CLAM/FTL}$; Table 3 shows the results. Columns contain, for each property, data about the proof plan and the proof (depth d , number of nodes #N, CPU Time in seconds), total CPU Time in seconds, and human time required to devise the solution (User, Proof, Tool time and total, in man-hours). The last two rows show time averages and totals.

Consider the “Averages” row. One can see that the average proof plan is 20 nodes deep and contains about 30 nodes in total (ratio: 0.66): proof plans are narrow and deep and not very large overall. This indicates that the proof planner chooses the right methods quite easily, also taking into account that basically no backtracking happens. In short, the abstract search space is tractable. On the other hand, the average *proof* is about 43 nodes deep and has 169 nodes in total (ratio: 0.25), which seems to suggest that there is a lot more “decoration” in a proof than in a proof plan — this agrees with the idea that the proof plan abstracts away much more than is allowed in a proof. Considering that here the search space is infinite and the object logic is non recursively enumerable, such a depth is remarkable. The plan is directing the search, which is the idea behind proof planning.

Proof planning time dominates over proof checking time by a factor of 3 to 2. This is sensible as well, since most of the “intelligence” of the system lies in the plan rather than in the proof, although the tactics in the methods can be rather involved, let alone requiring some degree of automation themselves. For instance, in some places the planner closes a branch assuming it can be closed at the object level too via propositional reasoning but then the object level theorem prover must exhaustively apply propositional reasoning in order to carry the proof to the end. Most of the time spent by the planner is required for reasoning on the shape of the formulae present in the sequent; in this case, higher order unification plays a leading role.

The whole set of benchmarks can be solved on a rather slow machine in something more than 10 minutes of CPU time, and the total human time required to set the machinery up was some 4 man-months full-time, assuming one man-month full-time is 160 man-hours; since this is a

novel approach, such an effort appears reasonable, since it also takes into account the time spent to debug a system which is still prototypical. To this extent, it is worth noting that there is definite dominance of Tool time over Proof time, and of Proof time over User time: detecting and fixing bugs is harder than designing methods and tactics, at least in the initial phase of the development of a novel approach.

Notice, lastly, that the human time reported in Table 3 is *not* time spent in human interaction with the system; once the methods and tactics have been devised, the process is automatic, if it runs to the end. Rather, one can think of User and Proof Time as time spent by the user in programming the planning and proof search of the system, and of Tool Time as time spent in debugging the system. This approach is quite different from standard interactive theorem proving.

5 Conclusions

In this thesis (*i*) we have devised sound, complete and easily mechanisable sequent calculi for a wide set of quantified modal logics; (*ii*) we have built an automated reasoning framework for **FOLTL** based upon Proof Planning; and lastly (*iii*) we have shown experimental results showing that, at least in one significant case-study, the approach is viable.

Related work. As far as contribution (*i*) is concerned, it is indeed not the first time that a labelled presentation of a wide spectrum of QMLs is given; the most remarkable piece of work so far is due to Viganò [31], who presented labelled Natural Deduction systems and sequent calculi for a subset of FO-axiomatisable QMLs, namely QMLs whose frame properties can be axiomatised by first-order *Horn clauses without equality* (what he calls *relational theories*). We extend his work by (*i*) lifting the restriction of Horn clauses, (*ii*) adding equality to the language of the frame properties, and (*iii*) showing a different way of proving soundness and completeness of such systems. On the same line, Negri and von Plato [26] have shown that axioms can be added to sequent calculi maintaining cut-freedom, through the representation of axioms as new rules of inference.

It is worth remarking here that the choice of labelled deduction is important, motivated by at least three reasons: (*i*) the explicit use of labels makes the presentation much more intuitive, in that it generates uniform sequent systems, (*ii*) it helps to keep reasoning about the properties of the frame separate from reasoning about logical formulae, thus aiding automated deduction, (*iii*) it gives rise to systems which can be inherently more powerful than unlabelled ones: see for instance [16], in which several unlabelled QMLs are proved incomplete with respect to their Kripke semantics (**QS4.2** on constant domains is a re-

Table 3: Experimental results.

Property	Proof plan			Proof				Human time			
	<i>d</i>	#N	Time	<i>d</i>	#N	Time		U	P	T	
Reach1	13	15	11	23	31	2	13	2	100	200	302
Reach2	19	21	24	66	92	7	31	1	10	1	12
Reach3	15	17	15	38	52	3	18	1	1	1	3
FO1	28	44	49	39	322	17	66	4	10	20	34
FO2	28	44	58	39	321	20	78	1	1	2	4
FO3	28	44	58	39	327	20	78	1	1	5	7
WU1	17	19	20	48	97	10	30	10	70	100	180
WU2	14	16	11	41	112	14	25	4	10	10	24
WU3	14	16	11	43	111	14	25	1	1	1	3
BCS'+WU1	17	19	21	57	112	13	34	1	1	1	3
BCS'+OCS	32	80	76	41	341	96	172	8	5	20	33
BCS'+WU3	14	16	11	47	110	20	31	20	10	10	40
Averages	20	29.6	30.4	43.4	169	19.7		3.5	18.3	30.9	
Totals			365			236	601	54	220	371	645

markable example — $C_{QS4.2}$ actually *is* sound and complete for it).

Contribution (ii) is totally novel. The development of a practical, hands-on integration between the proof planner λ CIAM and an object-level theorem prover had never been realised before, as far as we know.

Lastly, contribution (iii) shows that proof planning can effectively be applied to a problem arising in the Formal Methods community. Although the approach is not yet mature to be deployed among the tools for infinite-state formal methods, it represents a remarkable case-study for proof planning itself.

Concluding remarks. The thesis was defended at the School of Informatics, University of Edinburgh (UK), in January, 2005 and accepted with minor revisions. Parts of the work outlined here appear in [5, 6, 7, 8, 9]. The thesis is about to appear as a volume of the series “Research Perspectives in Logic”, edited by Polimetrica, Milan (Italy). The thesis is available from the Edinburgh University Research Archive at the URL <http://www.era.lib.ed.ac.uk/handle/1842/753>.

Acknowledgements. The author thanks his Ph.D. supervisor, Dr. Alan Smaill of the School of Informatics, University of Edinburgh, for his invaluable support throughout the work. The work was partially supported by EPSRC.

REFERENCES

[1] David Basin, Sean Matthews, and Luca Vigano. Labelled propositional modal logics: Theory and prac-

tice. *Journal of Logic and Computation*, 7(6):685–717, 1997.

- [2] A. Bundy. Proof planning. In B. Drabble, editor, *Proceedings of the 3rd International Conference on AI Planning Systems, (AIPS) 1996*, pages 261–267, 1996. also available as DAI Research Report 886.
- [3] Muffy Calder, Mario Kolberg, Evan H. Magill, and Stephan Reiff-Marganiec. Feature interaction: A critical review and considered forecast. *Computer Networks*, 2002.
- [4] Muffy Calder and Alice Miller. Feature interaction detection by pairwise analysis of LTL properties. Submitted to FMSD - Formal Methods in Software Development; still unpublished at the time of writing (January 2005), April 2002.
- [5] C. Castellini and A. Smaill. A modular, tactic-based approach to first-order temporal theorem proving. In *Proceedings of ICTL 2000*, Dresden, Germany, 2000.
- [6] C. Castellini and A. Smaill. Tactic-based theorem proving in first-order modal and temporal logics. In E. Giunchiglia and F. Massacci, editors, *Proceedings of IJCAR WS10: Issues in the Design and Experimental Evaluation of Systems for Modal and Temporal Logics*, TR DII 14/01. University of Siena, June 2001.
- [7] C. Castellini and A. Smaill. Proof planning for feature interactions: A preliminary report. In M. Baaz and A. Voronkov, editors, *Proceedings of LPAR 2002*, LNAI 2514, pages 102–114. Springer, October 2002.

- [8] C. Castellini and A. Smaill. A systematic presentation of quantified modal logics. *Logic Journal of the IGPL*, 10(6):571–599, 11 2002. Also available as Informatics Research Report EDI-INF-RR-0150, University of Edinburgh.
- [9] C. Castellini and A. Smaill. Proof planning for first-order temporal logic. In *proceedings of CADE, 20th International Conference on Automated Deduction, Tallinn (Estonia)*, 2005. To appear.
- [10] Claudio Castellini. *Automated Reasoning in Quantified Modal and Temporal Logics*. PhD thesis, School of Informatics, University of Edinburgh (Scotland), 2005.
- [11] A. Degtyarev and A. Voronkov. Equality reasoning in sequent-based calculi. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 10, pages 611–706. Elsevier Science, 2001.
- [12] F. J. Cantu, A. Bundy, A. Smaill, and D. Basin. Experiments in automating hardware verification using inductive proof planning. In M. Srivas and A. Camilleri, editors, *First international conference on formal methods in computer-aided design*, volume 1166 of *Lecture Notes in Computer Science*, pages 94–108, Palo Alto, CA, USA, November 1996. Springer Verlag.
- [13] D. Gabbay. *Labelled Deductive Systems*, volume 1. Oxford University Press, 1996.
- [14] J. Gallier. *Logic for Computer Science*. Harper & Row, New York, 1986.
- [15] Gerhard Gentzen. Investigations into logical deductions. In M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen (1969)*, pages 68–131. North-Holland Publishing Co., Amsterdam, 1969 edition, 1935.
- [16] Silvio Ghilardi. Incompleteness results in Kripke semantics. *The Journal of Symbolic Logic*, 56(2):517–538, June 1991.
- [17] Nancy Griffeth, Ralph Blumenthal, Jean-Charles Gregoire, and Tadashi Ohta. A feature interaction benchmark for the first feature interaction detection contest. *Computer Networks (Amsterdam, Netherlands: 1999)*, 32(4):389–418, April 2000.
- [18] Jaakko Hintikka. *Knowledge and Belief*. Cornell University Press, Ithaca, New York, 1962.
- [19] Ian Hodkinson, Frank Wolter, and Michael Zakaryashev. Monodic fragments of first-order temporal logics: 2000–2001 A.D. *Lecture Notes in Computer Science*, 2250:1–23, 2001.
- [20] G. P. Huet. A unification algorithm for typed lambda-calculus. note de travail A 055, Institut de Recherche d’Informatique et d’Automatique, March 1974.
- [21] S. Kanger. A simplified proof method for elementary logic. In *Computer Programming And Formal Systems, Studies in Logic*, pages 87–93. North-Holland Publ. Co., Amsterdam, 1963.
- [22] Saul A. Kripke. Semantical analysis of modal logic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963.
- [23] Dale Miller. A proposal for modules in lambda-prolog. In *Extensions of Logic Programming*, pages 206–221, 1993.
- [24] Dale Miller. λ Prolog: An introduction to the language and its logic. Unpublished as yet., 1998.
- [25] 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.
- [26] S. Negri and J. Von Plato. Cut elimination in the presence of axioms. *Bulletin of Symbolic Logic*, 4(4):418–435, Dec 1998.
- [27] J. D. C. Richardson, A. Smaill, and I. Green. System description: proof planning in higher-order logic with Lambda-Clam. In Claude Kirchner and Hélène Kirchner, editors, *15th International Conference on Automated Deduction*, volume 1421 of *Lecture Notes in Artificial Intelligence*, pages 129–133, Lindau, Germany, July 1998.
- [28] Alessandra Russo. Generalising propositional modal logic using labelled deductive systems. In F. Baader and K. U. Schulz, editors, *Frontiers of Combining Systems: Proceedings of the 1st International Workshop, Munich (Germany)*, Applied Logic, pages 57–74. Kluwer Academic Publishers, March 1996.
- [29] J.R. Shoenfield. *Mathematical logic*. Addison-Wesley, 1970.
- [30] Johan van Benthem. Correspondence theory. In D. Gabbay and F. Guenther, 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.
- [31] Luca Viganò. *Labelled Non-Classical Logics*. Kluwer Academic Publishers, Dordrecht, 2000.