Lemma 1 Ltd.
2nd Floor
31A Chain St.
Reading
Berks
RG1 2HX

Ramsey's Theorem in ProofPower (Draft)

Abstract

This paper is concerned with a ProofPower-HOL proof of the finite exponent 2 version of Ramsey's Theorem. A listing of the HOL theory in which the theorem has been proved is given together with a brief discussion of the proof in comparison with the proofs using NQTHM and Nuprl discussed in a CLI technical report by David Basin and Matt Kaufmann, [2].

Version:	2.11
Date:	29th September 1992; Last update: 31st January 2010.
Reference:	PPDOC/WRK043
Pages:	13

Prepared by:	Roger Bishop Jones
	Rob Arthan
Tel:	+44 118 958 4409
E-Mail:	rbjones@rbjones.com
	rda@lemma-one.com

1 INTRODUCTION

The CLI technical report, [2], uses the finite exponent 2 version of Ramsey's Theorem as a benchmark to compare the NQTHM (Boyer-Moore) and Nuprl theorem provers. In order to compare ICL's ProofPower-HOL theorem prover with NQTHM and Nuprl, this theorem has been formulated and proved using ProofPower-HOL. As [2] shows of NQTHM and Nuprl, the comparison indicates that ProofPower-HOL has its own particular advantages and disadvantages arising from the philosophies behind its design.

2 THE HOL LOGIC

Like NQTHM and Nuprl, if not more so, the ProofPower-HOL prover uses a well-defined and well-understood logic, namely a polymorphic variant of Church's simple type theory originally due to M.J.C. Gordon.

The logic is classical and may be viewed as polymorphically typed formulation of a set theory with the axiom of choice (but not the axiom of replacement). The language is at heart just the simply-typed λ-calculus with a polymorphic type system similar in spirit to that of Standard ML. On top of this core language, derived syntactic constructions, such as *let*-expressions and set comprehensions, are defined giving a surface syntax which is very like that of conventional formal mathematics.

The logic provides conservative extension mechanisms for introducing new types and constants. These mechanisms take as parameters theorems which justify the conservativeness of the extension. For example, on the basis of a theorem of the form $\vdash \exists x \bullet p[x/x]$ we may conservatively introduce a new constant c satisfying the defining axiom $\vdash p[c/x]$. The HOL theorem prover carefully distinguishes the conservative defining axioms from other axioms in the database of theories which it manages. The axiomatic basis for the theory described in the present document comprises only the agreed axioms for HOL as defined in [3] (which is based on the formulation of HOL used in the Cambridge HOL system described in [4]).

The logic supports the polymorphism in that it allows polymorphic theorems to be instantiated to particular types as required. For example, the polymorphic theorem:

$$\vdash \forall\ x{:}'a \bullet\ \mathit{Size}\ \{x\} = 1$$

giving the size of a singleton set may be used as a rewrite rule to prove any of the following theorems:

$$\vdash \mathit{Size}\ \{1\} = 1$$
$$\vdash \mathit{Size}\ \{(1,2)\} = 1$$
$$\vdash \mathit{Size}\ \{\mathit{Cons}\ 1\ []\} = 1$$

in which the type variable $'a$ has been instantiated to natural numbers, pairs of natural numbers and lists of natural numbers respectively.

3 THE THEOREM PROVER

The ProofPower-HOL theorem prover actually provides support both for developing specifications in HOL as well as carrying out proofs. The definitions in terms of which we formulate Ramsey's theorem constitute such a specification and are given in section 4 below.

The system is normally used to develop specifications or proofs interactively, an off-the-shelf windowing system and editor supplying the user-interface. The command language (or metalanguage) with which the user controls the system is an extension of the strongly-typed functional programming language, Standard ML. Like Nuprl, the system is constructed following the LCF paradigm, [1], which affords a good degree of assurance in the integrity of the implementation of the logic. The ML type discipline is used to guarantee that all inference is ultimately channelled through one of a small number of primitive inference rules implemented as ML functions computing values of an abstract datatype, THM, of theorems.

The extensions to ML essentially comprise a macro-processing front-end which allow convenient entry of object language constructs such as specification paragraphs or terms needed as parameters to proof rules.

While the primitive inference system is defined solely using forward inference rules, it is possible to construct powerful proof procedures which work either forwards or backwards. Most proof development work is carried out using the so-called subgoal package which provides an interactive environment for developing proofs in a goal-oriented (i.e. backwards) style. At the detailed level, both forwards and backwards steps may be taken within the subgoal package.

4 THE SPECIFICATION

In order to state the Ramsey theorem, the graph-theoretic notions involved in its statement are required. ProofPower-HOL comes supplied with a library of theories[1]including a theory of relations viewed as sets of pairs. There is also a theory of finiteness, which has the theory of relations as an ancestor.The theory in which we work is therefore set up as a child of the theory of finiteness using the following metalanguage commands:

```
SML
open_theory"fin_thms";
new_theory"ramsey";
```

First we define the notion of a symmetric graph. The following specification paragraph introduces a new constant *symg* which is the (polymorphic) set of all pairs (V, E) in which V is a set and E is a binary relation in V. Note that the set type constructor $\mathbb{P}$ is written as a post-fix operator and that the polymorphic type $'a\ \mathbb{P}$ comprises the set of all subsets of $'a$ finite or infinite. $\leftrightarrow$ is defined in the

[1]In ProofPower a "theory", is analogous to a module in programming language terms. A theory comprises sets of axioms, definitions, and theorems, together with additional information, e.g., the fixity of operators. As with the theory developed in this document, the set of axioms is typically empty, indicating that the theory is a conservative extension of its ancestors. The ProofPower system manages a database of theories organised as a directed acyclic graph.

theory of relations as an infix operator which assigns to two sets, A and B say, the set $A \leftrightarrow B$ of binary relations between A and B.

HOL Constant

$$\boldsymbol{symg} : (\prime a\ \mathbb{P} \times (\prime a \leftrightarrow \prime a))\mathbb{P}$$

$$\forall\ (V,E) \bullet\ (V,E) \in symg$$

$$\Leftrightarrow E \in (V \leftrightarrow V) \land \forall\ x\ y \bullet\ (x,y) \in E \Leftrightarrow (y,x) \in E$$

Now we define an infix operator *clique_of* such that $C\ clique\_of\ (V,\ E)$ is true iff. C is a clique in the graph (V, E). The term $(C\ \times\ C)\backslash(Id\ C)$ is the set-theoretic difference of the complete relation $(C\ \times\ C)$ on C and the identity relation $Id\ C$ on C. Thus the second conjunct asserts that any pair of distinct elements of C are related by E.

SML

```
declare_infix(300, "clique_of");
```

HOL Constant

$$\$\boldsymbol{clique\_of}\ :\ \prime a\ \mathbb{P} \rightarrow (\prime a\ \mathbb{P} \times (\prime a \leftrightarrow \prime a)) \rightarrow BOOL$$

$$\forall\ C\ (V,E) \bullet\ C\ clique\_of\ (V,E)$$

$$\Leftrightarrow C \subseteq V \land (C \times C)\backslash(Id\ C) \subseteq E$$

In a similar way we define an infix operator *indep_of* such that $C\ indep\_of\ (V,\ E)$ is true iff. C is an independent subset in the graph (V, E):

SML

```
declare_infix(300, "indep_of");
```

HOL Constant

$$\$\boldsymbol{indep\_of}\ :\ \prime a\ \mathbb{P} \rightarrow (\prime a\ \mathbb{P} \times (\prime a \leftrightarrow \prime a)) \rightarrow BOOL$$

$$\forall\ C\ (V,E) \bullet\ C\ indep\_of\ (V,E)$$

$$\Leftrightarrow C \subseteq V \land (C \times C) \cap E \subseteq Id\ C$$

That completes the specification. A listing of the theory introduced by this specification (together with the theorems which have been proved about it) may be found in Appendix A. The listing is incorporated using a simple interface between the ProofPower tool and the LaTeX document preparation system.

5 PROOF STRATEGY

Ramsey's theorem, in the finite exponent 2 case we are interested in, is the following conjecture:

$$
\begin{array}{l}
\forall\ a\ b \bullet \exists n \bullet \\
\forall\ (V,E) \bullet \\
(V,E) \in \mathit{symg} \wedge V \in \mathit{Finite} \wedge \#V \geq n \Rightarrow \\
\qquad (\exists\ C \bullet C\ \mathit{clique\_of}\ (V,E) \wedge \#C = a \\
\qquad \vee\ C\ \mathit{indep\_of}\ (V,E) \wedge \#C = b)
\end{array}
$$

That is to say, for any natural numbers, a and b there is an n such that any finite symmetric graph with at least n vertices contains either a clique with a vertices or an independent set with b vertices.

An informal proof of this conjecture is given in [2], and the proof strategy we take is essentially to follow that proof, with the slight difference that it turns out to be valid and easier to start the induction at 0 rather than at 2. The proof consists of 13 lemmas of which one is a simple generality about finite sets which was not available in the desired form in the theory of finite sets (which was under development during the production of the present proof). 7 of the other lemmas are trivialities about cliques and independent subsets etc. which are needed in several places in the proof. The other 5 lemmas are essentially dedicated to reducing the complexity of the proof of the conjecture by breaking it into manageable pieces. The theory listing in appendix A shows the 13 lemmas and the Ramsey theorem itself.

In more detail, what was done was to analyse the inductive step in [2] to get an understanding of the constructions it makes. The main content of the inductive step is captured in lemma 12. This is itself broken down into an argument about the size of a set given as the union of three subsets one of which is a singleton (lemma 11) and into the construction of two subgraphs one or other of which either contains or can be extended by one element to contain the desired clique or independent set (lemmas 9 and 10 justify the construction). Lemma 13 gives the base case of the induction.

6 THE PROOF

Space does not permit the inclusion of the full proof here. We just show the preamble and the proof of the first lemma.

We prepare for the proof by assigning ML names to the specifications and setting the proof context[2]:

```
SML
val symg_def = get_spec⌜symg⌝;
val clique_of_def = get_spec⌜$clique_of⌝;
val indep_of_def = get_spec⌜$indep_of⌝;
set_pc"hol";
```

[2] A *proof context* in ProofPower is a named collection of settings for global parameters which affect the operation of many of the more commonly used proof procedures. It allows domain-specific information to be made available to these proof procedures in a way which is uniform for the implementer and convenient for the user.

We invoke the subgoal package to begin the proof of the first lemma as follows.

SML

```
set_goal([], ⌜
        ∀(V, E); U•
        (V, E) ∈ symg ∧ U ⊆ V ⇒ (U, (U × U) ∩ E) ∈ symg
⌝);
```

We first rewrite with definitions to reduce an assertion about symmetric graphs into a set-theoretic assertion:

SML

```
a(rewrite_tac[symg_def, ×_def, ↔_def]);
```

This gives the following output:

ProofPower Output

```
Tactic produced 1 subgoal:

(* *** Goal "" *** *)

(* ?⊢ *)  ⌜∀ (V, E) U
          • (E ∈ ℙ {(v, w)|v ∈ V ∧ w ∈ V} ∧ (∀ x y• (x, y) ∈ E ⇔ (y, x) ∈ E))
              ∧ U ⊆ V
            ⇒ {(v, w)|v ∈ U ∧ w ∈ U} ∩ E ∈ ℙ {(v, w)|v ∈ U ∧ w ∈ U}
              ∧ (∀ x y
              • (x, y) ∈ {(v, w)|v ∈ U ∧ w ∈ U} ∩ E
                  ⇔ (y, x) ∈ {(v, w)|v ∈ U ∧ w ∈ U} ∩ E)⌝
```

We now attack this with a standard tactic for breaking down a goal using a proof context *hol*1 which is good for proving set-theoretic results by element level reasoning.

SML

```
a(REPEAT(PC_T "hol1" strip_tac));
```

This gives the following output[3] which indicates that we have 6 assumptions on the basis of which we must prove $(y,\ x)\ \in\ E$.

[3] In fact the output in this case contains a little more than what is shown. There are actually two subgoals which turn out to be the same modulo renaming of variables. The subgoal package informs us of this, but only requires us to prove one of the subgoals. This phenomenon is not very common but it is a useful labour-saver when it occurs.

ProofPower Output

```
(* *** Goal "1" *** *)

(*  6 *)  ⌜∀ x• x ∈ E ⇒ x ∈ {(v, w)|v ∈ V ∧ w ∈ V}⌝
(*  5 *)  ⌜∀ x y• (x, y) ∈ E ⇔ (y, x) ∈ E⌝
(*  4 *)  ⌜∀ x• x ∈ U ⇒ x ∈ V⌝
(*  3 *)  ⌜x ∈ U⌝
(*  2 *)  ⌜y ∈ U⌝
(*  1 *)  ⌜(x, y) ∈ E⌝

(* ?⊢ *)  ⌜(y, x) ∈ E⌝
```

We now use a tactic which works by forward chaining using the assumptions to complete the proof:

SML

```
a(asm_fc_tac[]);
```

Resulting in the following output:

ProofPower Output

```
Tactic produced 0 subgoals:
Current and main goal achieved
```

We save the theorem in the theory for future reference as follows:

SML

```
val lemma1 = save_pop_thm"lemma1";
```

This results in the following output from the metalanguage compiler indicating that we have succeeded in computing a theorem and assigning it to the metalanguage variable $lemma1$.

ProofPower Output

```
val lemma1 = ⊢ ∀(V, E) U• (V, E) ∈ symg ∧ U ⊆ V ⇒ (U, (U × U) ∩ E) ∈ symg : THM
```

In fact the above proof has been broken down into smaller steps than an experienced user of the system would typically use. In particular the first two steps would commonly be entered on one line using a tactic combinator to compose the tactics to avoid seeing the unstripped form of the rewritten goal.

7 STATISTICS

The total specification and proof effort was about 10 hours. The first author prepared and debugged the specification (by attempting various parts of the proof) in about an hour and the second author

proved the result in about 9 hours, of which at least one was spent proving conjectures which turned out to be irrelevant to the main proof.

The proof script contains 225 tactic applications and about 2,600 tokens. The replay time for the script is 8.0 minutes[4] and the script causes 47,489 primitive inference steps to be invoked.

We give the above token count for comparison with [2]. It is however felt that the simple time taken to type the script in is not a big inhibitor. Also several parts of the script were rapidly generated by cut-and-paste methods, since the symmetries in the problem mean that many proofs are rather similar (e.g. the proof of lemma 10 took about 5 minutes to find once the proof of lemma 9 was available).

[4]The timing was taken using a Sun SPARCStation 1+ with 12 megabytes of memory.

A THE THEORY ramsey

A.1 Parents

fin_thms

A.2 Constants

symg	$('a\ SET \times ('a \times 'a)\ SET)\ SET$
$\$\boldsymbol{clique\_of}$	$'a\ SET \to 'a\ SET \times ('a \times 'a)\ SET \to BOOL$
$\$\boldsymbol{indep\_of}$	$'a\ SET \to 'a\ SET \times ('a \times 'a)\ SET \to BOOL$

A.3 Fixity

Right Infix 300:

clique_of ***indep_of***

A.4 Definitions

symg $\vdash \forall\ (V,\ E)$
$\bullet\ (V,\ E) \in symg$
$\Leftrightarrow E \in V \leftrightarrow V \wedge (\forall\ x\ y\bullet\ (x,\ y) \in E \Leftrightarrow (y,\ x) \in E)$

clique_of $\vdash \forall\ C\ (V,\ E)$
$\bullet\ C\ clique\_of\ (V,\ E) \Leftrightarrow C \subseteq V \wedge (C \times C) \setminus Id\ C \subseteq E$

indep_of $\vdash \forall\ C\ (V,\ E)$
$\bullet\ C\ indep\_of\ (V,\ E) \Leftrightarrow C \subseteq V \wedge (C \times C) \cap E \subseteq Id\ C$

A.5 Theorems

lemma1 $\vdash \forall\ (V,\ E)\ U$
$\bullet\ (V,\ E) \in symg \wedge U \subseteq V \Rightarrow (U,\ (U \times U) \cap E) \in symg$

lemma2 $\vdash \forall\ (V,\ E)\ U\ C$
$\bullet\ U \subseteq V \wedge C\ clique\_of\ (U,\ (U \times U) \cap E)$
$\Rightarrow C\ clique\_of\ (V,\ E)$

lemma3 $\vdash \forall\ (V,\ E)\ U\ C$
$\bullet\ U \subseteq V \wedge C\ indep\_of\ (U,\ (U \times U) \cap E)$
$\Rightarrow C\ indep\_of\ (V,\ E)$

lemma4 $\vdash \forall\ (V,\ E)\ U\ C$
$\bullet\ C \subseteq U \wedge C\ indep\_of\ (V,\ E) \Rightarrow C\ indep\_of\ (U,\ E)$

lemma5 $\vdash \forall\ (V,\ E)\ U\ C$
$\bullet\ U \subseteq V \wedge C\ clique\_of\ (U,\ E) \Rightarrow C\ clique\_of\ (V,\ E)$

lemma6 $\vdash \forall\ (V,\ E)\ U\ C$
$\bullet\ U \subseteq V \wedge C\ indep\_of\ (U,\ E) \Rightarrow C\ indep\_of\ (V,\ E)$

lemma7 $\vdash (\{\},\ \{\}) \in symg$

lemma8 $\vdash \forall\ a\bullet\ a \in Finite \Rightarrow (\#\ a = 0 \Leftrightarrow a = \{\})$

lemma9 $\vdash \forall\ (V,\ E)\ v\ S\ i\ j$
- $(let\ R = \{x | x \in V \setminus \{v\} \land (v,\ x) \in E\}$
  $in\ (V,\ E) \in symg$
  $\land\ V \in Finite$
  $\land\ v \in V$
  $\land\ S \subseteq R$
  $\land\ (S\ clique\_of\ (V,\ (R \times R) \cap E) \land \#\ S = i$
  $\lor\ S\ indep\_of\ (V,\ (R \times R) \cap E) \land \#\ S = j)$
  $\Rightarrow (\exists\ S'$
  - $S'\ clique\_of\ (V,\ E) \land \#\ S' = i + 1$
    $\lor\ S'\ indep\_of\ (V,\ E) \land \#\ S' = j))$

lemma10 $\vdash \forall\ (V,\ E)\ v\ S\ i\ j$
- $(let\ R = \{x | x \in V \setminus \{v\} \land \neg\ (v,\ x) \in E\}$
  $in\ (V,\ E) \in symg$
  $\land\ V \in Finite$
  $\land\ v \in V$
  $\land\ S \subseteq R$
  $\land\ (S\ clique\_of\ (V,\ (R \times R) \cap E) \land \#\ S = i$
  $\lor\ S\ indep\_of\ (V,\ (R \times R) \cap E) \land \#\ S = j)$
  $\Rightarrow (\exists\ S'$
  - $S'\ clique\_of\ (V,\ E) \land \#\ S' = i$
    $\lor\ S'\ indep\_of\ (V,\ E) \land \#\ S' = j + 1))$

lemma11 $\vdash \forall\ a\ b\ c\ x\ m\ n$
- $a \in Finite \land \#\ a \geq m + n \land a = \{x\} \cup b \cup c$
  $\Rightarrow \#\ b \geq m \lor \#\ c \geq n$

lemma12 $\vdash \forall\ a\ b\ m\ n$
- $0 < m$
  $\land\ 0 < n$
  $\land\ (\forall\ (V,\ E)$
  - $(V,\ E) \in symg \land V \in Finite \land \#\ V \geq m$
    $\Rightarrow (\exists\ C$
    - $C\ clique\_of\ (V,\ E) \land \#\ C = a$
      $\lor\ C\ indep\_of\ (V,\ E) \land \#\ C = b + 1))$

  $\land\ (\forall\ (V,\ E)$
  - $(V,\ E) \in symg \land V \in Finite \land \#\ V \geq n$
    $\Rightarrow (\exists\ C$
    - $C\ clique\_of\ (V,\ E) \land \#\ C = a + 1$
      $\lor\ C\ indep\_of\ (V,\ E) \land \#\ C = b))$

  $\Rightarrow (\forall\ (V,\ E)$
  - $(V,\ E) \in symg \land V \in Finite \land \#\ V \geq m + n$
    $\Rightarrow (\exists\ C$
    - $C\ clique\_of\ (V,\ E) \land \#\ C = a + 1$
      $\lor\ C\ indep\_of\ (V,\ E) \land \#\ C = b + 1))$

lemma13 $\vdash \forall\ (V,\ E)$- $\{\}\ clique\_of\ (V,\ E) \land \{\}\ indep\_of\ (V,\ E)$

fin_exp_2_ramsey_thm
$\vdash \forall\ a\ b$
- $\exists\ n$
  - $\forall\ (V,\ E)$

$$\bullet\ (V,\ E) \in \mathit{symg} \land V \in \mathit{Finite} \land \#\ V \geq n$$
$$\Rightarrow (\exists\ C$$
$$\bullet\ C\ \mathit{clique\_of}\ (V,\ E) \land \#\ C = a$$
$$\lor\ C\ \mathit{indep\_of}\ (V,\ E) \land \#\ C = b)$$

B INDEX

C REFERENCES

[1] Michael J.C. Gordon, Arthur J. Milner, and Christopher P. Wadsworth. *Edinburgh LCF. Lecture Notes in Computer Science. Vol. 78.* Springer-Verlag, 1979.

[2] *The Boyer-Moore Prover and Nuprl: An Experimental Comparison.* David Basin and Matt Kaufmann, CLI, 1990.

[3] DS/FMU/IED/SPC003. *HOL Formalised: Deductive System.* R.D. Arthan, Lemma 1 Ltd., `http://www.lemma-one.com`.

[4] *The HOL System: Description.* SRI International, 4 December 1989.