A Simple Formalization and Proof for the Mutilated Chess Board

LAWRENCE C. PAULSON, *Computer Laboratory, University of Cambridge, England. E-mail: lcp@cl.cam.ac.uk*

Abstract

The impossibility of tiling the mutilated chess board has been formalized and verified using Isabelle. The formalization is concise because it is expressed using inductive definitions. The proofs are straightforward except for some lemmas concerning finite cardinalities. This exercise is an object lesson in choosing a good formalization: one at the right level of abstraction.

Keywords: mutilated chess board, inductive definitions, Isabelle

1 Introduction

A chess board can be tiled by 32 dominoes, each covering two squares. If two diagonally opposite squares are removed, can the remaining 62 squares be tiled by dominoes? No. Each domino covers a white square and a black square, so a tiled area must have equal numbers of both colours. The mutilated board cannot be tiled because the two removed squares have the same colour (Fig. 1).

The mutilated chess board problem has stood as a challenge to the automated reasoning community since McCarthy [8] posed it in 1964. Robinson [15] outlines the history of the problem, citing Max Black as its originator.

Anybody can grasp the argument instantly, but even formalizing the problem seems hard, let alone proving it. McCarthy has recently renewed his challenge, publishing a formalization that he claims is suitable for any 'heavy duty set theory' prover [9].

Formalizations like this destroy the simplicity of the original problem. They typically define complicated predicates to recognize objects. To recognize dominoes, a predicate checks whether its argument contains two adjacent squares. Subramanian defines *adjacent* by comparing co-ordinates [17, 18]:

 © Oxford University Press

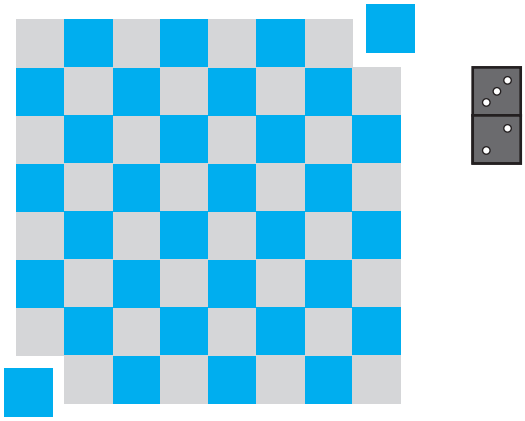

FIG. 1. The Mutilated Chess Board

```
(defn adjp (s1 s2)
  (or (and (equal (car s1) (car s2))
           (equal (plus 1 (cdr s1)) (cdr s2)))
      (and (equal (cdr s1) (cdr s2))
           (equal (plus 1 (car s1)) (car s2)))
      ))
```

Subramanian makes other definitions whose combined effect is to recognize a list of non-overlapping dominoes and to compute the region covered. McCarthy's formalization has a similar flavour, though posed in the language of sets. It is concise but formidable.

An alternative is to express the notion of tiling by an inductive definition. It is concise and nearly as clear as the informal problem statement. It provides an induction principle that is well-suited to proving the desired theorem.

2 Mathematical development

First we must make the intuitive argument rigorous. A *tile* is a set, regarded as a set of positions. A *tiling* (using a given set A of tiles) is defined inductively to be either the empty set or the union of a tiling with a tile $a \in A$ disjoint from it. Thus, a tiling is a finite union of disjoint tiles drawn from A.

This view is abstract and general. None of the sets have to be finite; we need not specify what positions are allowed. Now let us focus on chess boards.

A *square* is a pair (i, j) of natural numbers: an *even* (or *white*) square if $i + j$ is even and otherwise an *odd* (or *black*) square.

Let $\mathsf{lessThan}(n) = \{i \mid i < n\}$. (In set theory $n = \{i \mid i < n\}$ by definition, but some people find that confusing.) The Cartesian product

$\mathsf{lessThan}(8) \times \mathsf{lessThan}(8)$ expresses a 64-square chess board; it is the union of 8 disjoint rows of the form $\{i\} \times \mathsf{lessThan}(8)$ for $i = 0, \ldots, 7$.

A *domino* is a tile of the form $\{(i, j), (i, j+1)\}$ or $\{(i, j), (i+1, j)\}$. Since tilings are finite, we can use induction to prove that every tiling using dominoes has equally many even squares as odd squares.

Every row of the form $\{i\} \times \mathsf{lessThan}(2n)$ can be tiled using dominoes. As the union of two disjoint tilings is itself a tiling, every matrix of the form $\mathsf{lessThan}(2m) \times \mathsf{lessThan}(2n)$ can be tiled using dominoes. So every $2m \times 2n$ matrix has as many even squares as odd squares. (Informal treatments never bother to prove that a chess board has equal numbers of black and white squares.) The diagonally opposite squares $(0, 0)$ and $(2m-1, 2n-1)$ are both even; removing them results in a set that has fewer even squares than odd squares. No such set, including the mutilated chess board, can be tiled using dominoes.

3 The formal definitions

Isabelle [12] is a generic proof assistant, supporting many logics including ZF set theory and higher-order logic. I have done this exercise using both Isabelle/ZF and Isabelle/HOL. The definitions and proofs are similar in both systems. My formalization should be easy to mechanize in theorem provers that support inductive definitions, such as Coq [4] and HOL [5]. Higher-order logic simplifies the presentation slightly; type checking eliminates premises such as $i \in \mathsf{nat}$.

Figure 2 presents the theory file for the Isabelle/HOL version. It makes all the definitions needed for the chess board problem: tilings, dominoes and square colourings. Note that Suc is the successor function (mapping n to $n+1$) and that #2 denotes the number two. Keywords of the theory file syntax are underlined for clarity.

An inductive definition specifies the desired introduction rules. An Isabelle package defines the appropriate least fixedpoint and proves the introduction and induction rules [11]. The set of tilings using a set A of tiles is defined inductively. The Isabelle syntax appearing in Fig 2 expresses these two rules:

$$\emptyset \in \mathsf{tiling}(A) \qquad \frac{a \in A \quad t \in \mathsf{tiling}(A) \quad a \cap t = \emptyset}{a \cup t \in \mathsf{tiling}(A)}$$

Why does `tiling` have type `('a set)set ⇒ ('a set)set`? The symbol `'a` is a *type variable*. Isabelle/HOL is *polymorphic*: the type-checker automatically replaces each type variable by the type required by the context.

```
Mutil = Main +

consts     tiling :: "('a set) set ⇒ ('a set) set"
inductive "tiling A"
  intrs
    empty  "{} ∈ tiling A"
    Un     "⟦a ∈ A;  t ∈ tiling A;  a ∩ t = {}⟧
            ⟹ a ∪ t ∈ tiling A"

consts     domino  :: "(nat*nat)set set"
inductive "domino"
  intrs
    horiz  "{(i, j), (i, Suc j)} ∈ domino"
    vertl  "{(i, j), (Suc i, j)} ∈ domino"

constdefs
  coloured  :: "nat ⇒ (nat*nat)set"
   "coloured b == {(i,j). (i+j) mod #2 = b}"

end
```

FIG. 2. Isabelle/HOL Definitions of Dominoes and Tilings

In effect, `'a` is the type of squares. Each tile is a set of squares, so it has type `'a set`. The set A of tiles therefore has type `('a set)set`, as does the set of tilings generated by A.

The set of dominoes is inductively defined too. The Isabelle syntax expresses two introduction rules:

$$\{(i, j), (i, j+1)\} \in \mathsf{domino} \qquad \{(i, j), (i+1, j)\} \in \mathsf{domino}$$

The 'induction' here is trivial, but no matter, this definition is easy to use. It is declarative. Contrast it with the version appearing in Sect. 1, which is a piece of Lisp code. The constant `domino` has type `(nat*nat)set set` because it is a set of sets of pairs of natural numbers.

Figure 2 defines coloured b as set of squares having colour b. Formally, it is the set of even squares if $b = 0$ and the odd squares if $b = 1$. The set lessThan(n) is predefined in Isabelle/HOL to be $\{i \mid i < n\}$.

4 A primer on rule induction

You are probably familiar with 'mathematical induction' and with structural induction over lists and similar datatypes. An inductive definition gives rise to a principle sometimes known as *rule induction*. Given the definition of tiling, Isabelle generates the corresponding induction rule, shown here using

mathematical notation:

$$\frac{z \in \mathsf{tiling}(A) \qquad P(\emptyset) \qquad \begin{array}{c}[a \in A \quad t \in \mathsf{tiling}(A) \quad P(t) \quad a \cap t = \emptyset] \\ \vdots \\ P(a \cup t)\end{array}}{P(z)}$$

In English, *a property that is closed under the introduction rules for* $\mathsf{tiling}(A)$ *holds for all elements of* $\mathsf{tiling}(A)$. Induction is sound because $\mathsf{tiling}(A)$ is the *least* set closed under those rules. (This is why it is called rule induction.) In the inductive step, we are given an arbitrary tile $a \in A$ and tiling $t \in \mathsf{tiling}(A)$ that are disjoint ($a \cap t = \emptyset$) and satisfy the induction hypothesis $P(t)$.

A trivial rule induction proves that if each $a \in A$ is a finite set then so is $\mathsf{tiling}(A)$. Here $P(z)$ is the property $\mathsf{finite}(z)$. By induction, it suffices to show

- $\mathsf{finite}(\emptyset)$, which is trivial,
- and that $a \in A$ and $\mathsf{finite}(t)$ imply $\mathsf{finite}(a \cup t)$. This holds because we have assumed $\mathsf{finite}(a)$ for all $a \in A$.

The induction rule for dominoes has no induction hypothesis. A property holds for all dominoes provided it holds for the two possibilities given in the inductive definition. In the last two premises, i and j are arbitrary natural numbers.

$$\frac{z \in \mathsf{domino} \quad P(\{(i,j),(i,j+1)\}) \quad P(\{(i,j),(i+1,j)\})}{P(z)}$$

It is time for a harder example of induction. Let us prove that the union of two disjoint tilings is itself a tiling:

$$\frac{t \in \mathsf{tiling}(A) \quad u \in \mathsf{tiling}(A) \quad t \cap u = \emptyset}{t \cup u \in \mathsf{tiling}(A)}$$

This induction must be set up with care. Here $P(z)$ is the formula

$$u \in \mathsf{tiling}(A) \to (t \cap u = \emptyset \to t \cup u \in \mathsf{tiling}(A)) \tag{4.1}$$

The induction formula must be an implication because the induction variable, t, also occurs in $t \cap u = \emptyset$.

By induction on t there are two cases.

```
Goal "t ∈ tiling A ⟹
      u ∈ tiling A → t ∩ u = {} → t ∪ u ∈ tiling A";
by (etac tiling.induct 1);
by (simp_tac (simpset() addsimps [Un_assoc]) 2);
by Auto_tac;
```

FIG. 3. Isabelle/HOL Proof: the Union of Disjoint Tilings is a Tiling

- *Base case*. Putting $t = \emptyset$ in the formula (4.1), we must show

$$u \in \mathsf{tiling}(A) \to (\emptyset \cap u = \emptyset \to \emptyset \cup u \in \mathsf{tiling}(A))$$

  This is trivial because $\emptyset \cup u = u \in \mathsf{tiling}(A)$.
- *Inductive step*. We assume disjoint sets $a \in A$ and $t \in \mathsf{tiling}(A)$, as usual. The induction hypothesis is simply (4.1). We must show

$$u \in \mathsf{tiling}(A) \to ((a \cup t) \cap u = \emptyset \to (a \cup t) \cup u \in \mathsf{tiling}(A))$$

  To prove this implication, we assume $u \in \mathsf{tiling}(A)$ and $(a \cup t) \cap u = \emptyset$, which yields $a \cap u = \emptyset$ and $t \cap u = \emptyset$. From the induction hypothesis (4.1) we have $t \cup u \in \mathsf{tiling}(A)$. Since a is disjoint from both t and u, we may add it to the tiling $t \cup u$ to obtain $a \cup (t \cup u) \in \mathsf{tiling}(A)$.

5 The mechanical proofs

The Isabelle proofs offer few surprises. Finite cardinalities are tricky to reason about, as I have noted in previous work [14]. I needed a couple of hours to find a machine proof that a domino consists of one even square and one odd square. Another trouble spot was to prove that removing elements from a finite set reduces its cardinality: $|A - \{x\}| < |A|$ if A is finite and $x \in A$. One outcome of this exercise is a collection of general theorems about remainders and cardinality, which I have installed in Isabelle/HOL.

Apart from these trouble spots, the mechanized proof was straightforward. Developing the original ZF version took under 24 working hours. Excluding facts added to libraries, the (HOL) definitions and proof script occupy about 4400 bytes. They execute in 8.5 seconds on a 600MHz Pentium. Both figures are tiny, as suits this toy problem.

Figure 3 presents part of the script: the inductive proof outlined in the previous section. The script may be difficult to understand, but we see that proving this theorem requires little detail from the user. The `Goal` command supplies the theorem to be proved. The next line applies rule induction. Then

the simplifier (`simp_tac`) is called with an associativity theorem in order to replace $(a \cup t) \cup u$ by $a \cup (t \cup u)$. The rest of the proof is done by the automatic proof tactic, `Auto_tac`.

The full proof script, comprising 13 theorems, is Appendix A. Isabelle can display formulas using the fonts of X-symbol package [19], making formulas much more readable on-screen than they are in raw ASCII; I have edited the script to use similar symbols. Let us review the proofs informally.

5.1 On tiling chess boards

The first theorem has already been discussed in Sect. 4 and Fig. 3. We now develop a geometry of chess boards. The next two theorems (each proved by `Auto_tac`) relate `lessThan(Suc n)` and Cartesian products.

```
lessThan(Suc n) × B  =  ({n} × B) ∪ ((lessThan n) × B)
A × lessThan(Suc n)  =  (A × {n}) ∪ (A × (lessThan n))
```

Next comes a lemma, proved by `Auto_tac`, concerning singleton sets and Cartesian products. It makes a useful rewrite rule.

```
({i} × {n}) ∪ ({i} × {m}) = {(i,m), (i,n)}
```

The next two results state that *a row or matrix with an even number of columns can be tiled with dominoes*.

```
        {i} × lessThan(#2*n) ∈ tiling domino
(lessThan m) × lessThan(#2*n) ∈ tiling domino
```

These theorems apply to a standard 8×8 chess board, but not to a 9×9 one. The first theorem has a four-step proof, by induction on `n`. The simplifier massages `lessThan(#2 * Suc n)` into the union of a domino with the tiling given in the induction hypothesis. Then a tiling rule is applied explicitly. Finally, the automatic tactic (given the lemma proved above) finishes off. The second theorem has a trivial proof: induction over `m` followed by `Auto_tac`.

5.2 On colours and dominoes

Here is a simple fact about the squares in a tiling of a specified colour.

```
coloured b ∩ (insert (i,j) t) =
  (if (i+j) mod #2 = b then insert (i,j) (coloured b ∩ t)
                       else coloured b ∩ t)
```

Here insert x A denotes $\{x\} \cup A$. The b-coloured squares of $\{(i, j)\} \cup t$ comprise the b-coloured squares of t along with (i, j), if this square is coloured b. Although obvious, this fact is useful for rewriting. The proof is a one-liner: `Auto_tac`.

This fact is used to prove that a domino covers one square of each colour:

```
d ∈ domino ⟹
    (∃i j. coloured 0 ∩ d = {(i,j)}) ∧
    (∃m n. coloured 1 ∩ d = {(m,n)})
```

The proof is again simple. The first step is induction (really case analysis) on the domino. The automatic tactic finishes the proof, given a rewrite rule that reduces $(m + 1)$ mod n to m mod n.

5.3 On the cardinalities of some finite sets

For us, a domino is a two-element set of squares. Clearly all dominoes are finite, and a region tiled by dominoes is finite. Both proofs use induction followed by `Auto_tac`.

```
d ∈ domino        ⟹ finite d
t ∈ tiling domino ⟹ finite t
```

Most of the papers describing the chess board proof omit to mention that the board has finitely many squares. However, finiteness is crucial to the counting argument. (Infinite tiling problems are very different from finite ones. An infinite chess board can be tiled with dominoes even after one black square has been removed.)

Every set tiled by dominoes (such as an 8×8 chess board) contains equally many black squares as white ones. Here `card` is the cardinality function.

```
t ∈ tiling domino ⟹ card(coloured 0 ∩ t) = card(coloured 1 ∩ t)
```

This fact is also usually omitted from informal accounts, presumably because it is obvious. But its proof, six steps long, is not trivial. After applying induction, we use a fact proved above, namely that a domino covers one square of each colour. We are left having to show

```
card(insert sq0 (coloured 0 ∩ t)) = card(insert sq1 (coloured 1 ∩ t))
```

where `sq0` and `sq1` are the newly covered squares. The induction hypothesis is

```
card(coloured 0 ∩ t) = card(coloured 1 ∩ t).
```

Two proof steps show that the uses of `insert` add a square that was not already in the set. The result follows because both cardinalities increase by one.

5.4 Towards the main result

The main result presents some difficulties. Take the general case of removing any two white (even) squares, not necessarily in the corners.

```
⟦ t ∈ tiling domino;
  (i+j) mod #2 = 0;  (m+n) mod #2 = 0;
  {(i,j),(m,n)} ⊆ t ⟧
⟹ (t - {(i,j)} - {(m,n)}) ∉ tiling domino
```

In English, *removing two white squares from a region tiled with dominoes leaves a region that cannot be tiled*. The proof consists of five steps. The first simply assumes that the region can be tiled, for contradiction. Next we claim that there are fewer white squares than black, from which (step 3) we immediately obtain a contradiction. The last two steps prove the claim. It is surprisingly hard to prove that removing two elements from the set of white squares reduces its cardinality.

The main result is proved for any board with positive even dimensions. The mutilated board (less the two corners) cannot be tiled with dominoes.

```
t = lessThan(#2 * Suc m) × lessThan(#2 * Suc n)
⟹ t - {(0,0)} - {(Suc(#2*m), Suc(#2*n))} ∉ tiling domino
```

The proof applies the general theorem just discussed and discharges the first subgoal using a tiling lemma proved in Sect. 5.1. The rest falls to `Auto_tac`.

6 Related work and conclusions

In this note there is no space for a full literature review. Several efforts [2, 16, 18] are in the same spirit as the present work: the chess board is formalized and impossibility of tiling proved following the intuitive argument about colours. Other work has used exhaustive search or radical reformulations of the problem.

The Isabelle formalization compares favourably with the others. The definitions (Fig. 2) are concise, and in my view, easy to understand. The script is short: under 120 lines compared with over 500 for Subramanian [17]. (In terms of characters, which is more accurate, the ratio drops to 1:3.) According to McCarthy [9], Bancerek's mechanization [2] in Mizar requires 400 lines. Rudnicki's version [16] (also in Mizar) requires 300 lines. Andrews [1] reports a complex proof; it is not clear how much effort is needed to generate it.

When are inductive definitions appropriate? The choice is partly a matter of taste; published formalizations of the mutilated chess board show great diversity. Inductive definitions are ideal for finite constructions that allow non-determinism; the laying down of tiles fits that description precisely. The inductive definition plays the same role as Subramanian's finite state machine [18]. The initial state is the empty board; next states are obtained by adding disjoint tiles; properties that hold of all reachable states are proved by

induction. Giving an illegal input to the state machine sends it to an error state — a concept usually avoided with inductive definitions, since they describe only the legal constructions.

The finite state machine approach that Subramanian describes has been applied to substantial system verifications [10]. The inductive approach described above is an effective means of verifying cryptographic protocols [13]. Inductive definitions scale up to serious problems.

Acknowledgements. I learned of the expressiveness of inductive definitions through participation in the ESPRIT project 6453 TYPES, and especially through the work of Gérard Huet [6, 7]. John Harrison and anonymous referees commented on this paper.

References

[1] Peter B. Andrews and Matthew Bishop. On sets, types, fixed points, and checkerboards. In Pierangelo Miglioli, Ugo Moscato, Daniele Mundici, and Mario Ornaghi, editors, *Theorem Proving with Analytic Tableaux and Related Methods: 5th international workshop, TABLEAUX '96*, LNAI 1071, pages 1–15. Springer, 1996.

[2] Grzegorz Bancerek. The mutilated chessboard problem — checked by Mizar. In Boyer and Trybulec [3].

[3] Robert Boyer and Andrzej Trybulec, editors. *QED Workshop II*. On the World Wide Web at `http://www.mcs.anl.gov/qed/`, 1995.

[4] The Coq proof assistant. `http://coq.inria.fr/`, 2000.

[5] M. J. C. Gordon and T. F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge University Press, 1993.

[6] Gérard Huet. The Gallina specification language : A case study. In *Proceedings of 12th FST/TCS Conference, New Delhi*, LNCS 652. Springer, 1992.

[7] Gérard Huet. Residual theory in λ-calculus: A formal development. *Journal of Functional Programming*, 4(3):371–394, 1994.

[8] John McCarthy. A tough nut for proof procedures. Memo 16, Stanford Artificial Intelligence Project, July 1964.

[9] John McCarthy. The mutilated checkerboard in set theory. In Boyer and Trybulec [3].

[10] J Strother Moore. *Piton: A Mechanically Verified Assembly-Level Language*. Kluwer Academic Publishers, 1996.

[11] Lawrence C. Paulson. A fixedpoint approach to implementing (co)inductive definitions. In Alan Bundy, editor, *Automated Deduction — CADE-12 International Conference*, LNAI 814, pages 148–161. Springer, 1994.

[12] Lawrence C. Paulson. *Isabelle: A Generic Theorem Prover*. Springer, 1994. LNCS 828.

[13] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[14] Lawrence C. Paulson and Krzysztof Grąbczewski. Mechanizing set theory: Cardinal arithmetic and the axiom of choice. *Journal of Automated Reasoning*, 17(3):291–323, December 1996.

[15] J. A. Robinson. Formal and informal proofs. In Robert S. Boyer, editor, *Automated Reasoning: Essays in Honor of Woody Bledsoe*, pages 267–281. Kluwer Academic Publishers, 1991.

[16] Piotr Rudnicki. The mutilated checkerboard problem in the lightweight set theory of Mizar. `http://web.cs.ualberta.ca/~piotr/Mizar/Mutcheck`, November 1995.

[17] Sakthi Subramanian. A mechanically checked proof of the mutilated checkerboard theorem. `ftp://ftp.cs.utexas.edu/pub/boyer/nqthm/nqthm-1992/examples/subramanian/`, 1994.

[18] Sakthi Subramanian. An interactive solution to the $n \times n$ mutilated checkerboard problem. *Journal of Logic and Computation*, 6(4):573–598, 1996.

[19] Christoph Wedler. Emacs package “x-symbol”: Overview.
http://www.fmi.uni-passau.de/~wedler/x-symbol/, 2000.

A Full proof script

```
(*
The Mutilated Chess Board Problem, formalized inductively
*)

Addsimps (tiling.intrs @ domino.intrs);
AddIs    tiling.intrs;
```

Material discussed in Sect. 5.1

```
(** The union of two disjoint tilings is a tiling **)

Goal "t∈ tiling A ⟹ u∈ tiling A → t ∩ u = {} → t ∪ u ∈ tiling A";
by (etac tiling.induct 1);
by (simp_tac (simpset() addsimps [Un_assoc]) 2);
by Auto_tac;
qed_spec_mp "tiling_UnI";

AddIs [tiling_UnI];

(*** Chess boards ***)

Goalw [lessThan_def]
    "lessThan(Suc n) × B = ({n} × B) ∪ ((lessThan n) × B)";
by Auto_tac;
qed "Sigma_Suc1";

Goalw [lessThan_def]
    "A × lessThan(Suc n) = (A × {n}) ∪ (A × (lessThan n))";
by Auto_tac;
qed "Sigma_Suc2";

Addsimps [Sigma_Suc1, Sigma_Suc2];

Goal "({i} × {n}) ∪ ({i} × {m}) = {(i,m), (i,n)}";
by Auto_tac;
qed "sing_Times_lemma";

Goal "{i} × lessThan(#2*n) ∈ tiling domino";
by (induct_tac "n" 1);
by (ALLGOALS (asm_simp_tac (simpset() addsimps [Un_assoc RS sym])));
by (rtac tiling.Un 1);
by (auto_tac (claset(), simpset() addsimps [sing_Times_lemma]));
qed "dominoes_tile_row";

AddSIs [dominoes_tile_row];

Goal "(lessThan m) × lessThan(#2*n) ∈ tiling domino";
by (induct_tac "m" 1);
by Auto_tac;
qed "dominoes_tile_matrix";
```

Material discussed in Sect. 5.2

```
(*** "coloured" and Dominoes ***)

Goalw [coloured_def]
    "coloured b ∩ (insert (i,j) t) =
       (if (i+j) mod #2 = b then insert (i,j) (coloured b ∩ t)
                            else coloured b ∩ t)";
by Auto_tac;
qed "coloured_insert";
Addsimps [coloured_insert];

Goal "d ∈ domino ⟹ (∃i j. coloured 0 ∩ d = {(i,j)}) &
                     (∃m n. coloured 1 ∩ d = {(m,n)})";
by (etac domino.elim 1);
by (auto_tac (claset(), simpset() addsimps [mod_Suc]));
qed "domino_singletons";
```

Material discussed in Sect. 5.3

```
Goal "d ∈ domino ⟹ finite d";
by (etac domino.elim 1);
by Auto_tac;
qed "domino_finite";
Addsimps [domino_finite];


(*** Tilings of dominoes ***)

Goal "t ∈ tiling domino ⟹ finite t";
by (etac tiling.induct 1);
by Auto_tac;
qed "tiling_domino_finite";

Addsimps [tiling_domino_finite, Int_Un_distrib, Diff_Int_distrib];

Goal "t ∈ tiling domino ⟹ card(coloured 0 ∩ t) = card(coloured 1 ∩ t)";
by (etac tiling.induct 1);
by (dtac domino_singletons 2);
by Auto_tac;
(*this lemma tells us that both "inserts" are non-trivial*)
by (subgoal_tac "∀p C. C ∩ a = {p} → p ∉ t" 1);
by (Asm_simp_tac 1);
by (Blast_tac 1);
qed "tiling_domino_0_1";
```

Material discussed in Sect. 5.4

```
(*Final argument is surprisingly complex*)
Goal "⟦ t ∈ tiling domino;
        (i+j) mod #2 = 0;  (m+n) mod #2 = 0;
        {(i,j),(m,n)} ⊆ t ⟧
      ⟹ (t - {(i,j)} - {(m,n)}) ∉ tiling domino";
by (rtac notI 1);
by (subgoal_tac "card (coloured 0 ∩ (t - {(i,j)} - {(m,n)})) <
                 card (coloured 1 ∩ (t - {(i,j)} - {(m,n)}))" 1);
```

```
by (force_tac (claset(), HOL_ss addsimps [tiling_domino_0_1]) 1);
by (asm_simp_tac (simpset() addsimps [tiling_domino_0_1 RS sym]) 1);
by (asm_full_simp_tac
    (simpset() addsimps [coloured_def, card_Diff2_less]) 1);
qed "gen_mutil_not_tiling";

(*Apply the general theorem to the well-known case*)
Goal "t = lessThan(#2 * Suc m) × lessThan(#2 * Suc n)
      ⟹ t - {(0,0)} - {(Suc(#2*m), Suc(#2*n))} ∉ tiling domino";
by (rtac gen_mutil_not_tiling 1);
by (blast_tac (claset() addSIs [dominoes_tile_matrix]) 1);
by Auto_tac;
qed "mutil_not_tiling";
```

Received 11 September 2000