Fun With Tilings

Tobias Nipkow and Lawrence Paulson

December 14, 2021

Abstract

Tilings are defined inductively. It is shown that one form of mutilated chess board cannot be tiled with dominoes, while another one can be tiled with L-shaped tiles.

Sections 1 and 2 are by Paulson and described elsewhere [1]. Section 3 is by Nipkow and formalizes a well-known argument from the literature [2].

Please add further fun examples of this kind!

theory *Tilings* **imports** *Main* **begin**

1 Inductive Tiling

inductive-set
  tiling :: $'a$ *set set* ⇒ $'a$ *set set*
  for *A* :: $'a$ *set set* **where**
empty [*simp*, *intro*]: {} ∈ *tiling A* |
Un [*simp*, *intro*]: ⟦ *a* ∈ *A*; *t* ∈ *tiling A*; *a* ∩ *t* = {} ⟧
  ⟹ *a* ∪ *t* ∈ *tiling A*

lemma *tiling-UnI* [*intro*]:
  ⟦ *t* ∈ *tiling A*; *u* ∈ *tiling A*; *t* ∩ *u* = {} ⟧ ⟹ *t* ∪ *u* ∈ *tiling A*
apply (*induct set*: *tiling*)
apply (*auto simp add*: *Un-assoc*)
done

lemma *tiling-Diff1E*:
assumes *t*−*a* ∈ *tiling A* **and** *a* ∈ *A* **and** *a* ⊆ *t*
shows *t* ∈ *tiling A*
proof −
  from *assms*(*2*−*3*) **have** ∃ *r*. *t* = *r Un a* & *r Int a* = {}
    by (*metis Diff-disjoint Int-commute Un-Diff-cancel Un-absorb1 Un-commute*)
  thus *?thesis* **using** *assms*(*1*,*2*)
    by (*auto simp*:*Un-Diff*)
      (*metis Compl-Diff-eq Diff-Compl Diff-empty Int-commute Un-Diff-cancel*

Un-commute double-complement tiling.Un)
qed

lemma *tiling-finite*:
  assumes ⋀*a*. *a* ∈ *A* ⟹ *finite a*
  shows *t* ∈ *tiling A* ⟹ *finite t*
apply (*induct set*: *tiling*)
using *assms* **apply** *auto*
done

2 The Mutilated Chess Board Cannot be Tiled by Dominoes

The originator of this problem is Max Black, according to J A Robinson. It was popularized as the *Mutilated Checkerboard Problem* by J McCarthy.

inductive-set *domino* :: (*nat* × *nat*) *set set* **where**
horiz [*simp*]: {(*i*, *j*), (*i*, *Suc j*)} ∈ *domino* |
vertl [*simp*]: {(*i*, *j*), (*Suc i*, *j*)} ∈ *domino*

lemma *domino-finite*: *d* ∈ *domino* ⟹ *finite d*
by (*erule domino.cases*, *auto*)

declare *tiling-finite*[*OF domino-finite*, *simp*]

Sets of squares of the given colour

definition
  coloured :: *nat* ⇒ (*nat* × *nat*) *set* **where**
  coloured b = {(*i*, *j*). (*i* + *j*) *mod 2* = *b*}

abbreviation
  whites :: (*nat* × *nat*) *set* **where**
  whites ≡ *coloured 0*

abbreviation
  blacks :: (*nat* × *nat*) *set* **where**
  blacks ≡ *coloured* (*Suc 0*)

Chess boards

lemma *Sigma-Suc1* [*simp*]:
  {*0*..< *Suc n*} × *B* = ({*n*} × *B*) ∪ ({*0*..<*n*} × *B*)
by *auto*

lemma *Sigma-Suc2* [*simp*]:
  A × {*0*..< *Suc n*} = (*A* × {*n*}) ∪ (*A* × {*0*..<*n*})
by *auto*

lemma *dominoes-tile-row* [*intro*!]: {*i*} × {*0*..< *2*∗*n*} ∈ *tiling domino*

apply (*induct n*)
apply (*simp-all del:Un-insert-left add: Un-assoc* [*symmetric*])
done

lemma *dominoes-tile-matrix*: $\{0..<m\} \times \{0..<2*n\} \in$ *tiling domino*
by (*induct m*) *auto*

coloured and Dominoes

lemma *coloured-insert* [*simp*]:
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 simp add*: *coloured-def*)

lemma *domino-singletons*:
d ∈ *domino* ⟹
(∃ *i j*. *whites* ∩ *d* = {(*i*, *j*)}) ∧
(∃ *m n*. *blacks* ∩ *d* = {(*m*, *n*)})
apply (*erule domino.cases*)
apply (*auto simp add*: *mod-Suc*)
done

Tilings of dominoes

declare
Int-Un-distrib [*simp*]
Diff-Int-distrib [*simp*]

lemma *tiling-domino-0-1*:
t ∈ *tiling domino* ==> *card*(*whites* ∩ *t*) = *card*(*blacks* ∩ *t*)
apply (*induct set*: *tiling*)
apply (*drule-tac* [*2*] *domino-singletons*)
apply (*auto*)
apply (*subgoal-tac* ∀ *p C*. *C* ∩ *a* = {*p*} −−> *p* ∉ *t*)
— this lemma tells us that both "inserts" are non-trivial
apply (*simp* (*no-asm-simp*))
apply *blast*
done

Final argument is surprisingly complex

theorem *gen-mutil-not-tiling*:
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*
apply (*rule notI*)
apply (*subgoal-tac*
card (*whites* ∩ (*t* − {(*i*,*j*)} − {(*m*,*n*)})) <
card (*blacks* ∩ (*t* − {(*i*,*j*)} − {(*m*,*n*)})))

```
apply (force simp only: tiling-domino-0-1)
apply (simp add: tiling-domino-0-1 [symmetric])
apply (simp add: coloured-def card-Diff2-less)
done
```

Apply the general theorem to the well-known case

```
theorem mutil-not-tiling:
  t = {0..< 2 ∗ Suc m} × {0..< 2 ∗ Suc n}
   ==> t − {(0,0)} − {(Suc(2 ∗ m), Suc(2 ∗ n))} ∉ tiling domino
apply (rule gen-mutil-not-tiling)
apply (blast intro!: dominoes-tile-matrix)
apply auto
done
```

3 The Mutilated Chess Board Can be Tiled by Ls

Remove a arbitrary square from a chess board of size $2^n \times 2^n$. The result can be tiled by L-shaped tiles: . The four possible L-shaped tiles are obtained by dropping one of the four squares from $\{(x, y), (x + 1, y), (x, y + 1), (x + 1, y + 1)\}$:

```
definition L2 (x::nat) (y::nat) = {(x,y), (x+1,y), (x, y+1)}
definition L3 (x::nat) (y::nat) = {(x,y), (x+1,y), (x+1, y+1)}
definition L0 (x::nat) (y::nat) = {(x+1,y), (x,y+1), (x+1, y+1)}
definition L1 (x::nat) (y::nat) = {(x,y), (x,y+1), (x+1, y+1)}
```

All tiles:

```
definition Ls :: (nat ∗ nat) set set where
Ls ≡ { L0 x y | x y. True} ∪ { L1 x y | x y. True} ∪
     { L2 x y | x y. True} ∪ { L3 x y | x y. True}

lemma LinLs: L0 i j : Ls & L1 i j : Ls & L2 i j : Ls & L3 i j : Ls
by(fastforce simp:Ls-def)
```

Square $2^n \times 2^n$ grid, shifted by i and j:

```
definition square2 (n::nat) (i::nat) (j::nat) = {i..< 2^n+i} × {j..< 2^n+j}

lemma in-square2[simp]:
  (a,b) : square2 n i j ⟷ i≤a ∧ a<2^n+i ∧ j≤b ∧ b<2^n+j
by(simp add:square2-def)

lemma square2-Suc: square2 (Suc n) i j =
  square2 n i j ∪ square2 n (2^n + i) j ∪ square2 n i (2^n + j) ∪
  square2 n (2^n + i) (2^n + j)
by(auto simp:square2-def)

lemma square2-disj: square2 n i j ∩ square2 n x y = {} ⟷
  (2^n+i ≤ x ∨ 2^n+x ≤ i) ∨ (2^n+j ≤ y ∨ 2^n+y ≤ j) (is ?A = ?B)
proof−
```

```
  { assume ?B hence ?A by(auto simp:square2-def) }
  moreover
  { assume ¬ ?B
    hence (max i x, max j y) : square2 n i j ∩ square2 n x y by simp
    hence ¬ ?A by blast }
  ultimately show ?thesis by blast
qed
```

Some specific lemmas:

```
lemma pos-pow2: (0::nat) < 2^(n::nat)
by simp

declare nat-zero-less-power-iff[simp del] zero-less-power[simp del]

lemma Diff-insert-if: shows
  B ≠ {} ⟹ a:A ⟹ A − insert a B = (A−B − {a}) and
  B ≠ {} ⟹ a ~: A ⟹ A − insert a B = A−B
by auto

lemma DisjI1: A Int B = {} ⟹ (A−X) Int B = {}
by blast
lemma DisjI2: A Int B = {} ⟹ A Int (B−X) = {}
by blast
```

The main theorem:

```
theorem Ls-can-tile: i ≤ a ⟹ a < 2^n + i ⟹ j ≤ b ⟹ b < 2^n + j
  ⟹ square2 n i j − {(a,b)} : tiling Ls
proof(induct n arbitrary: a b i j)
  case 0 thus ?case by (simp add:square2-def)
next
  case (Suc n) note IH = Suc(1) and a = Suc(2−3) and b = Suc(4−5)
  hence a<2^n+i ∧ b<2^n+j ∨
       2^n+i≤a ∧ a<2^(n+1)+i ∧ b<2^n+j ∨
       a<2^n+i ∧ 2^n+j≤b ∧ b<2^(n+1)+j ∨
      2^n+i≤a ∧ a<2^(n+1)+i ∧ 2^n+j≤b ∧ b<2^(n+1)+j (is ?A|?B|?C|?D)
    by simp arith
  moreover
  { assume ?A
    hence square2 n i j − {(a,b)} : tiling Ls using IH a b by auto
    moreover have square2 n (2^n+i) j − {(2^n+i,2^n+j − 1)} : tiling Ls
      by(rule IH)(insert pos-pow2[of n], auto)
    moreover have square2 n i (2^n+j) − {(2^n+i − 1, 2^n+j)} : tiling Ls
      by(rule IH)(insert pos-pow2[of n], auto)
    moreover have square2 n (2^n+i) (2^n+j) − {(2^n+i, 2^n+j)} : tiling Ls
      by(rule IH)(insert pos-pow2[of n], auto)
    ultimately
    have square2 (n+1) i j − {(a,b)} − L0 (2^n+i − 1) (2^n+j − 1) ∈ tiling Ls
      using  a b ‹?A›
      by (clarsimp simp: square2-Suc L0-def Un-Diff Diff-insert-if)
```

```
    (fastforce intro!: tiling-UnI DisjI1 DisjI2 square2-disj[THEN iffD2]
            simp:Int-Un-distrib2)
} moreover
{ assume ?B
  hence square2 n (2^n+i) j − {(a,b)} : tiling Ls using IH a b by auto
  moreover have square2 n i j − {(2^n+i − 1,2^n+j − 1)} : tiling Ls
    by(rule IH)(insert pos-pow2[of n], auto)
  moreover have square2 n i (2^n+j) − {(2^n+i − 1, 2^n+j)} : tiling Ls
    by(rule IH)(insert pos-pow2[of n], auto)
  moreover have square2 n (2^n+i) (2^n+j) − {(2^n+i, 2^n+j)} : tiling Ls
    by(rule IH)(insert pos-pow2[of n], auto)
  ultimately
  have square2 (n+1) i j − {(a,b)} − L1 (2^n+i − 1) (2^n+j − 1) ∈ tiling Ls
    using a b ‹?B›
    by (simp add: square2-Suc L1-def Un-Diff Diff-insert-if le-diff-conv2)
      (fastforce intro!: tiling-UnI DisjI1 DisjI2 square2-disj[THEN iffD2]
            simp:Int-Un-distrib2)
} moreover
{ assume ?C
  hence square2 n i (2^n+j) − {(a,b)} : tiling Ls using IH a b by auto
  moreover have square2 n i j − {(2^n+i − 1,2^n+j − 1)} : tiling Ls
    by(rule IH)(insert pos-pow2[of n], auto)
  moreover have square2 n (2^n+i) j − {(2^n+i, 2^n+j − 1)} : tiling Ls
    by(rule IH)(insert pos-pow2[of n], auto)
  moreover have square2 n (2^n+i) (2^n+j) − {(2^n+i, 2^n+j)} : tiling Ls
    by(rule IH)(insert pos-pow2[of n], auto)
  ultimately
  have square2 (n+1) i j − {(a,b)} − L3 (2^n+i − 1) (2^n+j − 1) ∈ tiling Ls
    using a b ‹?C›
    by (simp add: square2-Suc L3-def Un-Diff Diff-insert-if le-diff-conv2)
      (fastforce intro!: tiling-UnI DisjI1 DisjI2 square2-disj[THEN iffD2]
            simp:Int-Un-distrib2)
} moreover
{ assume ?D
  hence square2 n (2^n+i) (2^n+j) −{(a,b)} : tiling Ls using IH a b by auto
  moreover have square2 n i j − {(2^n+i − 1,2^n+j − 1)} : tiling Ls
    by(rule IH)(insert pos-pow2[of n], auto)
  moreover have square2 n (2^n+i) j − {(2^n+i, 2^n+j − 1)} : tiling Ls
    by(rule IH)(insert pos-pow2[of n], auto)
  moreover have square2 n i (2^n+j) − {(2^n+i − 1, 2^n+j)} : tiling Ls
    by(rule IH)(insert pos-pow2[of n], auto)
  ultimately
  have square2 (n+1) i j − {(a,b)} − L2 (2^n+i − 1) (2^n+j − 1) ∈ tiling Ls
    using a b ‹?D›
    by (simp add: square2-Suc L2-def Un-Diff Diff-insert-if le-diff-conv2)
      (fastforce intro!: tiling-UnI DisjI1 DisjI2 square2-disj[THEN iffD2]
            simp:Int-Un-distrib2)
} moreover
have ?A ⟹ L0 (2^n + i − 1) (2^n + j − 1) ⊆ square2 (n+1) i j − {(a, b)}
```

using *a b* **by**(*simp add*:*L0-def*) *arith* **moreover**
have *?B* $\Longrightarrow$ *L1* (*2^n* + *i* − *1*) (*2^n* + *j* − *1*) ⊆ *square2* (*n*+*1*) *i j* − {(*a*, *b*)}
using *a b* **by**(*simp add*:*L1-def*) *arith* **moreover**
have *?C* $\Longrightarrow$ *L3* (*2^n* + *i* − *1*) (*2^n* + *j* − *1*) ⊆ *square2* (*n*+*1*) *i j* − {(*a*, *b*)}
using *a b* **by**(*simp add*:*L3-def*) *arith* **moreover**
have *?D* $\Longrightarrow$ *L2* (*2^n* + *i* − *1*) (*2^n* + *j* − *1*) ⊆ *square2* (*n*+*1*) *i j* − {(*a*, *b*)}
using *a b* **by**(*simp add*:*L2-def*) *arith*
ultimately show *?case* **by** *simp* (*metis LinLs tiling-Diff1E*)
qed

corollary *Ls-can-tile00*:
a < *2^n* $\Longrightarrow$ *b* < *2^n* $\Longrightarrow$ *square2 n 0 0* − {(*a*, *b*)} ∈ *tiling Ls*
by(*rule Ls-can-tile*) *auto*

end

References

[1] Lawrence C. Paulson. A simple formalization and proof for the mutilated chess board. *Logic J. of the IGPL*, 9(3), 2001.

[2] Velleman. *How to Prove it.* Cambridge University Press, 1994.