```
theory mutilated-actual
imports Main

begin

definition chessboard :: (nat × nat) set where
  chessboard = {..7} × {..7}

definition mchessboard :: (nat × nat) set where
  mchessboard = chessboard − {(0, 0), (7, 7)}

datatype domino =
  horizontal nat × nat
| vertical   nat × nat

fun covers :: domino ⇒ (nat × nat) set where
  covers (horizontal (i, j)) = {(i, j), (i+1, j)}
| covers (vertical (i, j)) = {(i, j), (i, j+1)}

definition white :: (nat × nat) set where
  white = {p. even (fst p + snd p)}

definition flip :: (nat × nat) ⇒ (nat × nat) where
  flip p = (7 − fst p, snd p)

theorem mutilated-chessboard:
  fixes s :: domino set
  assumes h1 : ∀ x ∈ s. ∀ y ∈ s. x ≠ y ⟶ covers x ∩ covers y = {} and
          h1a : finite s
  shows (⋃ x ∈ s. covers x) ≠ mchessboard
proof
  have bij-betw flip (chessboard ∩ white) (chessboard ∩ −white)
  proof −
    have aux1: ∀ x. x ∈ chessboard ⟶ flip x ∈ chessboard
      by (auto simp add: chessboard-def flip-def)
    have aux2: ∀ x ∈ chessboard. flip x ∈ white ⟷ x ∈ −white
      by (auto simp add: chessboard-def flip-def white-def)
    have aux3: inj-on flip (chessboard ∩ white)
      by (auto simp add: inj-on-def flip-def chessboard-def white-def)
    have aux4: ∀ x ∈ chessboard ∩ −white. x ∈ flip ‘ (chessboard ∩ white)
    proof −
      have aux4a: ∀ x ∈ chessboard. flip (flip x) = x
        by (auto simp add: flip-def chessboard-def)
      show ?thesis
        by (metis IntI Int-iff aux1 aux2 aux4a imageI)
    qed
    show ?thesis
      by (auto simp add: bij-betw-def aux1 aux2 aux3 aux4)
  qed
```

hence *h2*: *card* (*chessboard* ∩ *white*) = *card* (*chessboard* ∩ −*white*)
  using *bij-betw-same-card* **by** *blast*
have *h2a*: (*0*, *0*) ∈ *chessboard* ∩ *white* **and** *h2b*: (*7*, *7*) ∈ *chessboard* ∩ *white*
  by (*simp add*: *chessboard-def white-def*)+
have *h3*: *card* (*mchessboard* ∩ *white*) < *card* (*mchessboard* ∩ −*white*)
proof −
  have *aux5*: *mchessboard* ∩ *white* = *chessboard* ∩ *white* − {(*0*, *0*), (*7*, *7*)}
    by (*auto simp add*: *mchessboard-def*)
  have *aux6*: *mchessboard* ∩ −*white* = *chessboard* ∩ −*white*
    by (*auto simp add*: *mchessboard-def chessboard-def white-def*)
  have *aux7*: (*chessboard* ∩ *white* = *mchessboard* ∩ *white* ∪ {(*0*, *0*), (*7*, *7*)}) ∧
        (*mchessboard* ∩ *white*) ∩ {(*0*, *0*), (*7*, *7*)} = {}
    using *aux5 h2a h2b* **by** *auto*
  have *aux7a*: *finite chessboard* ∧ *finite mchessboard*
    by (*auto simp add*: *chessboard-def mchessboard-def*)
  have *aux7b*: (*0*, *0*) ∉ *insert* (*7*, *7*) (*mchessboard* ∩ *white*)
    using *h2a mchessboard-def* **by** *auto*
  have *aux7c*: (*7*, *7*) ∉ *mchessboard* ∩ *white*
    by (*simp add*: *aux5*)
  have *aux7d*: *card* (*insert* (*0*, *0*) (*insert* (*7*, *7*) (*mchessboard* ∩ *white*))) =
        Suc (*Suc* (*card* (*mchessboard* ∩ *white*)))
    using *aux7a aux7b aux7c* **by** *auto*
  have *aux8*: *card* (*chessboard* ∩ *white*) = *card* (*mchessboard* ∩ *white*) + *2*
    using *aux7* **by** (*auto simp add*: *aux7d*)
  show *?thesis*
    by (*auto simp add*: *aux5 aux6 aux8 h2* [*symmetric*])
qed
have *h9*: ∀ *x*. *card* (*covers x* ∩ *white*) = *card* (*covers x* ∩ −*white*)
proof −
  have *aux9a*: ∀ *a b*. (*a*, *b*) ∈ *white* ⟷ (*Suc a*, *b*) ∉ *white*
    by (*auto simp add*: *white-def*)
  have *aux9b*: ∀ *a b*. {(*a*, *b*), (*Suc a*, *b*)} ∩ *white* = {(*a*, *b*)} ∨
        {(*a*, *b*), (*Suc a*, *b*)} ∩ *white* = {(*Suc a*, *b*)}
   by (*metis Int-insert-left-if0 Int-insert-left-if1 aux9a inf-bot-right inf-commute*)
  have *aux9c*: ∀ *a b*. (*a*, *b*) ∈ *white* ⟷ (*a*, *Suc b*) ∉ *white*
    by (*auto simp add*: *white-def*)
  have *aux9d*: ∀ *a b*. {(*a*, *b*), (*a*, *Suc b*)} ∩ *white* = {(*a*, *b*)} ∨
        {(*a*, *b*), (*a*, *Suc b*)} ∩ *white* = {(*a*, *Suc b*)}
   by (*metis Int-insert-left-if1 Int-insert-right-if0 aux9c inf-bot-right inf-commute*)
  have *aux9e*: ∀ *x*. *card* (*covers x* ∩ *white*) = *1*
  proof
    fix *x*
    show *card* (*covers x* ∩ *white*) = *1*
      apply (*induction x*)
       apply *auto*
      using *aux9b card-Suc-eq* **apply** *force*
      using *aux9d card-Suc-eq* **by** *force*
  qed
  have *aux9f*: ∀ *a b*. {(*a*, *b*), (*Suc a*, *b*)} ∩ − *white* = {(*a*, *b*)} ∨

$\{(a,\ b),\ (Suc\ a,\ b)\} \cap -\ white = \{(Suc\ a,\ b)\}$

by (*metis Compl-disjoint Compl-iff Int-insert-left aux9a inf-commute inf-compl-bot-left1*)
have *aux9g*: $\forall\ a\ b.\ \{(a,\ b),\ (a,\ Suc\ b)\} \cap -\ white = \{(a,\ b)\} \vee$
$\{(a,\ b),\ (a,\ Suc\ b)\} \cap -\ white = \{(a,\ Suc\ b)\}$

by (*metis Compl-disjoint Compl-iff Int-insert-left-if0 Int-insert-left-if1 aux9c inf-right-idem*)
have *aux9h*: $\forall\ x.\ card\ (covers\ x \cap -white) = 1$
proof
fix *x*
show *card* (*covers x* $\cap$ − *white*) = *1*
apply (*induction x*)
apply *auto*
using *aux9f card-Suc-eq* **apply** *force*
using *aux9g card-Suc-eq* **by** *force*
qed
show *?thesis*
using *aux9e aux9h* **by** *simp*
qed
with *h1 h1a* **have** *h4*: $card\ ((\bigcup\ x \in s.\ covers\ x) \cap white) = card\ ((\bigcup\ x \in s.\ covers\ x) \cap -white)$
proof −
have *aux10e*: $\forall\ x.\ finite\ (covers\ x)$
proof
fix *x*
show *finite* (*covers x*)
apply (*cases x*)
by *auto*
qed
have *aux10f*: $\forall\ x.\ finite\ (covers\ x \cap white)$
using *aux10e* **by** *auto*
have *aux10g*: $\forall\ x \in s.\ \forall\ y \in s.\ x \neq y \longrightarrow (covers\ x \cap white) \cap (covers\ y \cap white) = \{\}$
using *h1* **by** *auto*
have *aux10h*: $\forall\ x \in s.\ \forall\ y \in s.\ x \neq y \longrightarrow (covers\ x \cap -\ white) \cap (covers\ y \cap -\ white) = \{\}$
using *h1* **by** *auto*
have *aux10a*: $(\bigcup\ x \in s.\ covers\ x) \cap white = (\bigcup\ x \in s.\ covers\ x \cap white)$
by *simp*
have *aux10b*: $(\bigcup\ x \in s.\ covers\ x) \cap -white = (\bigcup\ x \in s.\ covers\ x \cap -white)$
by *simp*
with *h1a aux10e aux10g* **have** *aux10c*:
$card\ ((\bigcup\ x \in s.\ covers\ x) \cap white) = (\sum\ x \in s.\ card\ (covers\ x \cap white))$
by (*smt aux10a card-UN-disjoint finite-Int sum.cong*)
with *h1 h1a aux10e aux10h* **have** *aux10d*:
$card\ ((\bigcup\ x \in s.\ covers\ x) \cap -\ white) = (\sum\ x \in s.\ card\ (covers\ x \cap -\ white))$
by (*smt aux10b card-UN-disjoint finite-Int sum.cong*)

show *?thesis*
by (*simp add*: *aux10c aux10d h9*)
qed
assume ($\bigcup$ $x \in s$. *covers* x) = *mchessboard*
with *h3 h4* **show** *False*
by *auto*
qed

end