

The Földes–Hammer Characterization of Split Graphs

Arthur Freitas Ramos David Barros Hulak
Ruy J. G. B. de Queiroz

October 6, 2026

Abstract

This entry formalizes the classical Földes–Hammer theorem characterizing finite split graphs as exactly the graphs with no induced copy of $2K_2$, C_4 , or C_5 . It builds on the AFP session `Undirected_Graph_Theory`.

1 Overview

A finite graph is split when its vertex set can be partitioned into a clique and an independent set. The main theorem of this entry is `foldes_hammer` in `Foldes_Hammer.thy`, which proves the equivalence between split graphs and the absence of induced $2K_2$, C_4 , and C_5 .

The forward direction is established by analyzing how any split partition excludes each forbidden configuration. The converse direction follows an extremal argument: choose a maximum clique whose complement minimizes the number of induced edges, orient any remaining outside edge according to nested nonneighbor sets in the clique, and swap one clique vertex with an outside vertex to derive a contradiction.

2 Sources

The entry follows the classical theorem due to Földes and Hammer [2] and standard graph-class background as presented by Brandstädt, Le, and Spinrad [1].

Contents

1 Overview

1

2 Sources

1

theory *Foldes-Hammer*

imports *Undirected-Graph-Theory.Undirected-Graphs-Root*
begin

context *fin-sgraph*
begin

definition *is-clique* :: 'a set $\Rightarrow$ bool **where**
is-clique $C \longleftrightarrow C \subseteq V \wedge \text{induced-edges } C = \text{all-edges } C$

definition *cliques* :: 'a set set **where**
cliques = $\{C. \text{is-clique } C\}$

definition *maximum-clique* :: 'a set $\Rightarrow$ bool **where**
maximum-clique $K \longleftrightarrow K \in \text{cliques} \wedge (\forall C \in \text{cliques}. \text{card } C \leq \text{card } K)$

definition *outside-edge-count* :: 'a set $\Rightarrow$ nat **where**
outside-edge-count $K = \text{card } (\text{induced-edges } (V - K))$

definition *is-split-partition* :: 'a set $\Rightarrow$ 'a set $\Rightarrow$ bool **where**
is-split-partition $C \ I \longleftrightarrow C \cap I = \{\} \wedge C \cup I = V \wedge \text{is-clique } C \wedge \text{is-independent-set } I$

definition *is-split-graph* :: bool **where**
is-split-graph $\longleftrightarrow (\exists C \ I. \text{is-split-partition } C \ I)$

definition *induces-2K2* :: 'a $\Rightarrow$ 'a $\Rightarrow$ 'a $\Rightarrow$ 'a $\Rightarrow$ bool **where**
induces-2K2 $a \ b \ c \ d \longleftrightarrow$
 $\text{distinct } [a, b, c, d] \wedge$
 $\text{vert-adj } a \ b \wedge \text{vert-adj } c \ d \wedge$
 $\neg \text{vert-adj } a \ c \wedge \neg \text{vert-adj } a \ d \wedge \neg \text{vert-adj } b \ c \wedge \neg \text{vert-adj } b \ d$

definition *induces-C4* :: 'a $\Rightarrow$ 'a $\Rightarrow$ 'a $\Rightarrow$ 'a $\Rightarrow$ bool **where**
induces-C4 $a \ b \ c \ d \longleftrightarrow$
 $\text{distinct } [a, b, c, d] \wedge$
 $\text{vert-adj } a \ b \wedge \text{vert-adj } b \ c \wedge \text{vert-adj } c \ d \wedge \text{vert-adj } d \ a \wedge$
 $\neg \text{vert-adj } a \ c \wedge \neg \text{vert-adj } b \ d$

definition *induces-C5* :: 'a $\Rightarrow$ 'a $\Rightarrow$ 'a $\Rightarrow$ 'a $\Rightarrow$ 'a $\Rightarrow$ bool **where**
induces-C5 $a \ b \ c \ d \ e \longleftrightarrow$
 $\text{distinct } [a, b, c, d, e] \wedge$
 $\text{vert-adj } a \ b \wedge \text{vert-adj } b \ c \wedge \text{vert-adj } c \ d \wedge \text{vert-adj } d \ e \wedge \text{vert-adj } e \ a \wedge$
 $\neg \text{vert-adj } a \ c \wedge \neg \text{vert-adj } a \ d \wedge \neg \text{vert-adj } b \ d \wedge \neg \text{vert-adj } b \ e \wedge \neg \text{vert-adj } c \ e$

definition *has-induced-2K2* :: bool **where**
has-induced-2K2 $\longleftrightarrow (\exists a \ b \ c \ d. \text{induces-2K2 } a \ b \ c \ d)$

definition *has-induced-C4* :: bool **where**

has-induced-C4 $\longleftrightarrow (\exists a\ b\ c\ d. \text{induces-C4}\ a\ b\ c\ d)$

definition *has-induced-C5* :: bool **where**

has-induced-C5 $\longleftrightarrow (\exists a\ b\ c\ d\ e. \text{induces-C5}\ a\ b\ c\ d\ e)$

definition *foldes-hammer-free* :: bool **where**

foldes-hammer-free $\longleftrightarrow \neg \text{has-induced-2K2} \wedge \neg \text{has-induced-C4} \wedge \neg \text{has-induced-C5}$

lemma *is-clique-alt*:

is-clique *C* $\longleftrightarrow C \subseteq V \wedge (\forall u \in C. \forall v \in C. u \neq v \longrightarrow \text{vert-adj}\ u\ v)$

proof

assume *H*: *is-clique* *C*

then have $C \subseteq V$ **by** (*simp add: is-clique-def*)

moreover have $\forall u \in C. \forall v \in C. u \neq v \longrightarrow \text{vert-adj}\ u\ v$

proof (*intro ballI impI*)

fix *u v*

assume *uC*: $u \in C$ **and** *vC*: $v \in C$ **and** *uv*: $u \neq v$

from *H* **have** *eq*: *induced-edges* *C* = *all-edges* *C*

by (*simp add: is-clique-def*)

have $\{u, v\} \in \text{all-edges}\ C$

using *uC vC uv* **by** (*auto simp: all-edges-alt*)

with *eq* **have** $\{u, v\} \in \text{induced-edges}\ C$ **by** *simp*

then show *vert-adj* *u v*

by (*auto simp: induced-edges-def vert-adj-def*)

qed

ultimately show $C \subseteq V \wedge (\forall u \in C. \forall v \in C. u \neq v \longrightarrow \text{vert-adj}\ u\ v)$

by *simp*

next

assume *H*: $C \subseteq V \wedge (\forall u \in C. \forall v \in C. u \neq v \longrightarrow \text{vert-adj}\ u\ v)$

then have *Csub*: $C \subseteq V$

and *adj*: $\forall u \in C. \forall v \in C. u \neq v \longrightarrow \text{vert-adj}\ u\ v$

by *auto*

have *induced-edges* *C* = *all-edges* *C*

proof (*rule subset-antisym*)

show *induced-edges* *C* $\subseteq$ *all-edges* *C*

using *induced-edges-alt* **by** *auto*

next

show *all-edges* *C* $\subseteq$ *induced-edges* *C*

proof

fix *e*

assume *eC*: $e \in \text{all-edges}\ C$

then obtain *u v* **where** *e*: $e = \{u, v\}$ **and** *uC*: $u \in C$ **and** *vC*: $v \in C$ **and**

uv: $u \neq v$

by (*auto simp: all-edges-alt*)

from *adj uC vC uv* **have** *vert-adj* *u v* **by** *blast*

then have $e \in E$

by (*simp add: e vert-adj-def*)

```

    moreover from  $eC$  have  $e \subseteq C$ 
      by (auto simp: all-edges-def)
    ultimately show  $e \in \text{induced-edges } C$ 
      by (simp add: induced-edges-def)
  qed
qed
with  $C_{\text{sub}}$  show  $\text{is-clique } C$ 
  by (simp add: is-clique-def)
qed

lemma clique-pair-adj:
  assumes  $\text{is-clique } C$   $u \in C$   $v \in C$   $u \neq v$ 
  shows  $\text{vert-adj } u \ v$ 
  using assms by (auto simp: is-clique-alt)

lemma clique-mono:
  assumes  $\text{is-clique } C$   $D \subseteq C$ 
  shows  $\text{is-clique } D$ 
  using assms by (auto simp: is-clique-alt)

lemma vert-adj-neg:
  assumes  $\text{vert-adj } u \ v$ 
  shows  $u \neq v$ 
proof
  assume  $uv\text{-eq}: u = v$ 
  from assms have  $\text{card2}: \text{card } \{u, v\} = 2$ 
    unfolding vert-adj-def using two-edges by blast
  from  $uv\text{-eq}$  have  $\text{card } \{u, v\} = 1$ 
    by simp
  with  $\text{card2}$  show False
    by simp
qed

lemma empty-clique:  $\text{is-clique } \{\}$ 
  by (simp add: is-clique-alt)

lemma singleton-clique:
  assumes  $v \in V$ 
  shows  $\text{is-clique } \{v\}$ 
  using assms by (auto simp: is-clique-alt)

lemma finite-cliques:  $\text{finite cliques}$ 
proof -
  have  $\text{cliques} \subseteq \text{Pow } V$ 
    by (auto simp: cliques-def is-clique-def)
  moreover have  $\text{finite } (\text{Pow } V)$ 
    using finV by simp
  ultimately show ?thesis
    by (rule finite-subset)

```

qed

lemma *cliques-nonempty*: $\{\} \in \text{cliques}$
by (*simp add: cliques-def empty-clique*)

lemma *maximum-clique-exists*:
obtains K **where** *maximum-clique* K
proof –
have fin : *finite* ($\text{card } \text{' cliques}$)
using *finite-cliques* **by** *simp*
have ne : $\text{card } \text{' cliques} \neq \{\}$
using *cliques-nonempty* **by** *auto*
let $?m = \text{Max } (\text{card } \text{' cliques})$
have m-in : $?m \in \text{card } \text{' cliques}$
using *Max-in*[*OF fin ne*] .
then obtain K **where** $K\text{-in}$: $K \in \text{cliques}$ **and** $K\text{-card}$: $\text{card } K = ?m$
by *auto*
have maxK : $\forall C \in \text{cliques}. \text{card } C \leq \text{card } K$
proof
fix C
assume $C\text{-in}$: $C \in \text{cliques}$
then have $\text{card } C \in \text{card } \text{' cliques}$
by *auto*
then have $\text{card } C \leq ?m$
using *Max-ge*[*OF fin*] **by** *blast*
then show $\text{card } C \leq \text{card } K$
by (*simp add: K-card*)
qed
have *maximum-clique* K
using $K\text{-in maxK}$ **by** (*simp add: maximum-clique-def*)
then show $?thesis$
using *that* **by** *blast*
qed

lemma *maximum-clique-finite*:
assumes *maximum-clique* K
shows *finite* K
using *assms finV finite-subset*
by (*auto simp: maximum-clique-def cliques-def is-clique-def*)

lemma *finite-induced-edges-subset-V*:
assumes $S \subseteq V$
shows *finite* (*induced-edges* S)
proof –
have finS : *finite* S
using *assms finV finite-subset* **by** *blast*
have *induced-edges* $S \subseteq \text{Pow } S$
by (*auto simp: induced-edges-def*)
moreover have *finite* ($\text{Pow } S$)

using *finS* by *simp*
 ultimately show *?thesis*
 by (rule *finite-subset*)
 qed

lemma *obtain-best-clique*:

obtains *K* where *maximum-clique K*
 and $\forall L. \text{maximum-clique } L \longrightarrow \text{outside-edge-count } K \leq \text{outside-edge-count } L$
 proof –
 let *?M* = {*K*. *maximum-clique K*}
 have *finM*: *finite ?M*
 proof (rule *finite-subset*[of *?M cliques*])
 show *?M* $\subseteq$ *cliques*
 by (auto simp: *maximum-clique-def*)
 show *finite cliques*
 using *finite-cliques* .
 qed
 have *neM*: *?M* $\neq$ {}
 using *maximum-clique-exists* by blast
 have *finImg*: *finite (outside-edge-count ‘ ?M)*
 using *finM* by simp
 have *neImg*: *outside-edge-count ‘ ?M* $\neq$ {}
 using *neM* by auto
 let *?m* = *Min (outside-edge-count ‘ ?M)*
 have *mIn*: *?m* $\in$ *outside-edge-count ‘ ?M*
 using *Min-in*[OF *finImg neImg*] .
 then obtain *K* where *Kmax*: *maximum-clique K* and *Kmin*: *outside-edge-count*
K = *?m*
 by auto
 have *Kbest*: $\forall L. \text{maximum-clique } L \longrightarrow \text{outside-edge-count } K \leq \text{outside-edge-count } L$
 using *Kmin finImg neImg*
 by (simp add: *Min-le*)
 show *?thesis*
 using that *Kmax Kbest* by blast
 qed

lemma *insert-clique-if-all-adj*:

assumes *C*: *is-clique C*
 and *vV*: *v* $\in$ *V*
 and *vnot*: *v* $\notin$ *C*
 and *adj*: $\forall u \in C. \text{vert-adj } u \ v$
 shows *is-clique (insert v C)*
 proof –
 from *C* have *Csub*: *C* $\subseteq$ *V*
 and *Cadj*: $\forall u \in C. \forall w \in C. u \neq w \longrightarrow \text{vert-adj } u \ w$
 by (auto simp: *is-clique-alt*)
 have *insert v C* $\subseteq$ *V*
 using *Csub vV* by auto

```

moreover have  $\forall u \in \text{insert } v \ C. \forall w \in \text{insert } v \ C. u \neq w \longrightarrow \text{vert-adj } u \ w$ 
proof (intro ballI impI)
  fix  $u \ w$ 
  assume  $uin: u \in \text{insert } v \ C$  and  $win: w \in \text{insert } v \ C$  and  $uw: u \neq w$ 
  show  $\text{vert-adj } u \ w$ 
  proof (cases  $u = v$ )
    case  $u\text{-eq-}v$ : True
      with  $win \ vnot \ uw$  have  $w \in C$  by auto
      then have  $\text{vert-adj } w \ v$ 
        using adj by simp
      then show ?thesis
        using  $u\text{-eq-}v$  by (simp add: vert-adj-sym)
    next
      case  $u\text{-ne-}v$ : False
      show ?thesis
      proof (cases  $w = v$ )
        case  $w\text{-eq-}v$ : True
          with  $uin \ vnot \ uw \ u\text{-ne-}v$  have  $u \in C$  by auto
          then have  $\text{vert-adj } u \ v$ 
            using adj by simp
          then show ?thesis
            using  $w\text{-eq-}v$  by simp
        next
          case  $w\text{-ne-}v$ : False
          from  $uin \ win \ u\text{-ne-}v \ w\text{-ne-}v$  have  $u \in C \ w \in C$  by auto
          with  $Cadj \ uw$  show ?thesis by blast
      qed
    qed
  qed
  ultimately show ?thesis
    by (simp add: is-clique-alt)
qed

```

```

lemma maximum-clique-outside-has-nonneighbor:
  assumes  $maxK$ : maximum-clique  $K$ 
    and  $vV$ :  $v \in V$ 
    and  $vnot$ :  $v \notin K$ 
  shows  $\exists u \in K. \neg \text{vert-adj } u \ v$ 
proof (rule ccontr)
  assume  $neg$ :  $\neg (\exists u \in K. \neg \text{vert-adj } u \ v)$ 
  then have  $all\text{-adj}$ :  $\forall u \in K. \text{vert-adj } u \ v$ 
    by auto
  from  $maxK$  have  $K\text{-clique}$ : is-clique  $K$ 
    by (auto simp: maximum-clique-def cliques-def)
  have  $ins\text{-clique}$ : is-clique (insert  $v \ K$ )
    using insert-clique-if-all-adj [OF  $K\text{-clique} \ vV \ vnot \ all\text{-adj}$ ] .
  then have  $ins\text{-in}$ : insert  $v \ K \in \text{cliques}$ 
    by (simp add: cliques-def)
  from  $maxK$  have  $bound$ :  $\forall C \in \text{cliques}. \text{card } C \leq \text{card } K$ 

```

```

    by (auto simp: maximum-clique-def)
  from maxK have Ksub:  $K \subseteq V$ 
    by (auto simp: maximum-clique-def cliques-def is-clique-def)
  then have finK: finite K
    using finV finite-subset by blast
  from ins-in bound have card (insert v K)  $\leq$  card K
    by blast
  with finK vnot show False
    by simp
qed

lemma split-graph-alt:
  is-split-graph  $\longleftrightarrow$  ( $\exists C. \text{is-clique } C \wedge \text{is-independent-set } (V - C)$ )
proof
  assume H: is-split-graph
  then obtain C I where sp: is-split-partition C I
    by (auto simp: is-split-graph-def)
  then have I = V - C
    by (auto simp: is-split-partition-def is-clique-def)
  with sp show  $\exists C. \text{is-clique } C \wedge \text{is-independent-set } (V - C)$ 
    by (auto simp: is-split-partition-def)
next
  assume H:  $\exists C. \text{is-clique } C \wedge \text{is-independent-set } (V - C)$ 
  then obtain C where C: is-clique C is-independent-set (V - C)
    by blast
  have is-split-partition C (V - C)
    using C by (auto simp: is-split-partition-def is-clique-def)
  then show is-split-graph
    by (auto simp: is-split-graph-def)
qed

lemma independent-pair-not-adj:
  assumes is-independent-set I  $u \in I$   $v \in I$ 
  shows  $\neg \text{vert-adj } u$   $v$ 
  using assms by (auto simp: is-independent-alt)

lemma split-partition-edge-hits-clique:
  assumes sp: is-split-partition C I
    and uv: vert-adj u v
  shows  $u \in C \vee v \in C$ 
proof (rule ccontr)
  assume notC:  $\neg (u \in C \vee v \in C)$ 
  from uv have uV:  $u \in V$  and vV:  $v \in V$ 
    using vert-adj-imp-inV by auto
  from sp uV vV notC have uI:  $u \in I$  and vI:  $v \in I$ 
    by (auto simp: is-split-partition-def)
  from sp have indep: is-independent-set I
    by (simp add: is-split-partition-def)
  from independent-pair-not-adj[OF indep uI vI] uv show False

```

by contradiction
qed

lemma *split-partition-nonedge-not-both-clique*:

assumes *sp*: *is-split-partition* *C I*

and *uV*: $u \in V$

and *vV*: $v \in V$

and *uv*: $u \neq v$

and *not-adj*: $\neg \text{vert-adj } u \ v$

shows $\neg (u \in C \wedge v \in C)$

proof

assume *uvC*: $u \in C \wedge v \in C$

then have *uC*: $u \in C$ and *vC*: $v \in C$

by *auto*

from *sp* have *clique*: *is-clique* *C*

by (*simp add: is-split-partition-def*)

from *clique-pair-adj*[*OF clique uC vC uv*] *not-adj* **show** *False*

by contradiction

qed

lemma *induces-2K2-inV*:

assumes *induces-2K2* *a b c d*

shows $a \in V \ b \in V \ c \in V \ d \in V$

using *assms vert-adj-imp-inV* **by** (*auto simp: induces-2K2-def*)

lemma *induces-C4-inV*:

assumes *induces-C4* *a b c d*

shows $a \in V \ b \in V \ c \in V \ d \in V$

using *assms vert-adj-imp-inV* **by** (*auto simp: induces-C4-def*)

lemma *induces-C5-inV*:

assumes *induces-C5* *a b c d e*

shows $a \in V \ b \in V \ c \in V \ d \in V \ e \in V$

using *assms vert-adj-imp-inV* **by** (*auto simp: induces-C5-def*)

lemma *split-partition-forbids-2K2*:

assumes *sp*: *is-split-partition* *C I*

and *H*: *induces-2K2* *a b c d*

shows *False*

proof –

from *H* have *ab*: *vert-adj* *a b* and *cd*: *vert-adj* *c d*

and *aV*: $a \in V$ and *bV*: $b \in V$ and *cV*: $c \in V$ and *dV*: $d \in V$

and *ac*: $a \neq c$ and *ad*: $a \neq d$ and *bc*: $b \neq c$ and *bd*: $b \neq d$

and *nac*: $\neg \text{vert-adj } a \ c$ and *nad*: $\neg \text{vert-adj } a \ d$ and *nbc*: $\neg \text{vert-adj } b \ c$ and

nbd: $\neg \text{vert-adj } b \ d$

using *induces-2K2-inV*[*OF H*] **by** (*auto simp: induces-2K2-def*)

have $a \in C \vee b \in C$

using *split-partition-edge-hits-clique*[*OF sp ab*] .

moreover have $c \in C \vee d \in C$

using *split-partition-edge-hits-clique*[*OF sp cd*] .
 moreover have $\neg (a \in C \wedge c \in C)$
 using *split-partition-nonedge-not-both-clique*[*OF sp aV cV ac nac*] .
 moreover have $\neg (a \in C \wedge d \in C)$
 using *split-partition-nonedge-not-both-clique*[*OF sp aV dV ad nad*] .
 moreover have $\neg (b \in C \wedge c \in C)$
 using *split-partition-nonedge-not-both-clique*[*OF sp bV cV bc nbc*] .
 moreover have $\neg (b \in C \wedge d \in C)$
 using *split-partition-nonedge-not-both-clique*[*OF sp bV dV bd nbd*] .
 ultimately show *False*
 by *blast*
 qed

lemma *split-partition-forbids-C4*:
 assumes *sp: is-split-partition C I*
 and *H: induces-C4 a b c d*
 shows *False*

proof –
 from *H* have *ab: vert-adj a b* and *bc: vert-adj b c* and *cd: vert-adj c d* and *da:*
vert-adj d a
 and *aV: a ∈ V* and *bV: b ∈ V* and *cV: c ∈ V* and *dV: d ∈ V*
 and *ac: a ≠ c* and *bd: b ≠ d*
 and *nac: ¬ vert-adj a c* and *nbd: ¬ vert-adj b d*
 using *induces-C4-inV[OF H]* by (*auto simp: induces-C4-def*)
 have $a \in C \vee b \in C$
 using *split-partition-edge-hits-clique*[*OF sp ab*] .
 moreover have $b \in C \vee c \in C$
 using *split-partition-edge-hits-clique*[*OF sp bc*] .
 moreover have $c \in C \vee d \in C$
 using *split-partition-edge-hits-clique*[*OF sp cd*] .
 moreover have $d \in C \vee a \in C$
 using *split-partition-edge-hits-clique*[*OF sp da*] .
 moreover have $\neg (a \in C \wedge c \in C)$
 using *split-partition-nonedge-not-both-clique*[*OF sp aV cV ac nac*] .
 moreover have $\neg (b \in C \wedge d \in C)$
 using *split-partition-nonedge-not-both-clique*[*OF sp bV dV bd nbd*] .
 ultimately show *False*
 by *blast*
 qed

lemma *split-partition-forbids-C5*:
 assumes *sp: is-split-partition C I*
 and *H: induces-C5 a b c d e*
 shows *False*

proof –
 from *H* have *ab: vert-adj a b* and *bc: vert-adj b c* and *cd: vert-adj c d* and *de:*
vert-adj d e and *ea: vert-adj e a*
 and *aV: a ∈ V* and *bV: b ∈ V* and *cV: c ∈ V* and *dV: d ∈ V* and *eV: e*
 $\in V$

and $ac: a \neq c$ **and** $ad: a \neq d$ **and** $bd: b \neq d$ **and** $be: b \neq e$ **and** $ce: c \neq e$
and $nac: \neg \text{vert-adj } a \ c$ **and** $nad: \neg \text{vert-adj } a \ d$ **and** $nbd: \neg \text{vert-adj } b \ d$ **and**
 $nbe: \neg \text{vert-adj } b \ e$ **and** $nce: \neg \text{vert-adj } c \ e$
using *induces-C5-inV[OF H]* **by** (*auto simp: induces-C5-def*)
have $a \in C \vee b \in C$
using *split-partition-edge-hits-clique[OF sp ab]* .
moreover have $b \in C \vee c \in C$
using *split-partition-edge-hits-clique[OF sp bc]* .
moreover have $c \in C \vee d \in C$
using *split-partition-edge-hits-clique[OF sp cd]* .
moreover have $d \in C \vee e \in C$
using *split-partition-edge-hits-clique[OF sp de]* .
moreover have $e \in C \vee a \in C$
using *split-partition-edge-hits-clique[OF sp ea]* .
moreover have $\neg (a \in C \wedge c \in C)$
using *split-partition-nonedge-not-both-clique[OF sp aV cV ac nac]* .
moreover have $\neg (a \in C \wedge d \in C)$
using *split-partition-nonedge-not-both-clique[OF sp aV dV ad nad]* .
moreover have $\neg (b \in C \wedge d \in C)$
using *split-partition-nonedge-not-both-clique[OF sp bV dV bd nbd]* .
moreover have $\neg (b \in C \wedge e \in C)$
using *split-partition-nonedge-not-both-clique[OF sp bV eV be nbe]* .
moreover have $\neg (c \in C \wedge e \in C)$
using *split-partition-nonedge-not-both-clique[OF sp cV eV ce nce]* .
ultimately show *False*
by *blast*
qed

lemma *split-graph-imp-folde-hammer-free*:
assumes *is-split-graph*
shows *folde-hammer-free*
proof –
obtain $C \ I$ **where** $sp: \text{is-split-partition } C \ I$
using *assms* **by** (*auto simp: is-split-graph-def*)
have $\neg \text{has-induced-}2K2$
using sp *split-partition-forbids-2K2* **by** (*auto simp: has-induced-2K2-def*)
moreover have $\neg \text{has-induced-}C4$
using sp *split-partition-forbids-C4* **by** (*auto simp: has-induced-C4-def*)
moreover have $\neg \text{has-induced-}C5$
using sp *split-partition-forbids-C5* **by** (*auto simp: has-induced-C5-def*)
ultimately show *?thesis*
by (*simp add: folde-hammer-free-def*)
qed

lemma *outside-adjacent-nonneighbor-sets-nested*:
assumes $fh: \text{folde-hammer-free}$
and $Kmax: \text{maximum-clique } K$
and $uO: u \in V - K$
and $vO: v \in V - K$

and uv : *vert-adj* u v
shows $\{x \in K. \neg \text{vert-adj } x \ u\} \subseteq \{x \in K. \neg \text{vert-adj } x \ v\} \vee$
 $\{x \in K. \neg \text{vert-adj } x \ v\} \subseteq \{x \in K. \neg \text{vert-adj } x \ u\}$
proof (*rule ccontr*)
let $?A = \{x \in K. \neg \text{vert-adj } x \ u\}$
let $?B = \{x \in K. \neg \text{vert-adj } x \ v\}$
assume *not-nested*: $\neg (?A \subseteq ?B \vee ?B \subseteq ?A)$
then obtain a b **where** aA : $a \in ?A$ **and** aB : $a \notin ?B$ **and** bB : $b \in ?B$ **and** bA :
 $b \notin ?A$
by *blast*
have aK : $a \in K$ **and** aNu : $\neg \text{vert-adj } a \ u$
using aA **by** *auto*
have bK : $b \in K$ **and** bNv : $\neg \text{vert-adj } b \ v$
using bB **by** *auto*
have av : *vert-adj* $a \ v$
using aB aK **by** *auto*
have bu : *vert-adj* $b \ u$
using bA bK **by** *auto*
have $ab\text{-}ne$: $a \neq b$
using aA bA **by** *auto*
have $uv\text{-}ne$: $u \neq v$
using *vert-adj-neg*[OF uv] .
have $Kclq$: *is-clique* K
using $Kmax$ **by** (*auto simp: maximum-clique-def cliques-def*)
have ab : *vert-adj* $a \ b$
using *clique-pair-adj*[OF $Kclq$ aK bK $ab\text{-}ne$] .
have vu : *vert-adj* $v \ u$
using uv **by** (*simp add: vert-adj-sym*)
have ub : *vert-adj* $u \ b$
using bu **by** (*simp add: vert-adj-sym*)
have ba : *vert-adj* $b \ a$
using ab **by** (*simp add: vert-adj-sym*)
have vNb : $\neg \text{vert-adj } v \ b$
using bNv **by** (*simp add: vert-adj-sym*)
have $dist$: *distinct* [a, v, u, b]
using aK bK uO vO $ab\text{-}ne$ $uv\text{-}ne$ **by** *auto*
have $induces\text{-}C_4$ $a \ v \ u \ b$
using $dist$ av vu ub ba aNu vNb
by (*simp add: induces- C_4 -def*)
then have $has\text{-}induced\text{-}C_4$
by (*auto simp: has-induced- C_4 -def*)
with fh **show** *False*
by (*simp add: foldes-hammer-free-def*)
qed

lemma *outside-adjacent-smaller-nonneighbor-set-singleton*:
assumes fh : *foldes-hammer-free*
and $Kmax$: *maximum-clique* K
and uO : $u \in V - K$

```

and  $vO$ :  $v \in V - K$ 
and  $uv$ :  $\text{vert-adj } u \ v$ 
and  $AB$ :  $\{x \in K. \neg \text{vert-adj } x \ u\} \subseteq \{x \in K. \neg \text{vert-adj } x \ v\}$ 
obtains  $a$  where  $\{x \in K. \neg \text{vert-adj } x \ u\} = \{a\}$  and  $a \in K$  and  $\neg \text{vert-adj } a \ u$ 
and  $\neg \text{vert-adj } a \ v$ 
proof –
  have  $uV$ :  $u \in V$  and  $u \notin K$ 
  using  $uO$  by auto
obtain  $a$  where  $aK$ :  $a \in K$  and  $aNu$ :  $\neg \text{vert-adj } a \ u$ 
  using maximum-clique-outside-has-nonneighbor[ $OF \ Kmax \ uV \ u \notin K$ ] by blast
have  $aA$ :  $a \in \{x \in K. \neg \text{vert-adj } x \ u\}$ 
  using  $aK \ aNu$  by auto
have  $aB$ :  $a \in \{x \in K. \neg \text{vert-adj } x \ v\}$ 
  using  $AB \ aA$  by blast
have  $aNv$ :  $\neg \text{vert-adj } a \ v$ 
  using  $aB$  by auto
have  $Aeq$ :  $\{x \in K. \neg \text{vert-adj } x \ u\} = \{a\}$ 
proof (rule subset-antisym)
  show  $\{x \in K. \neg \text{vert-adj } x \ u\} \subseteq \{a\}$ 
  proof
    fix  $b$ 
    assume  $bA$ :  $b \in \{x \in K. \neg \text{vert-adj } x \ u\}$ 
    show  $b \in \{a\}$ 
    proof (rule ccontr)
      assume  $bneq$ :  $b \notin \{a\}$ 
      then have  $ab-ne$ :  $a \neq b$ 
      by auto
      have  $bK$ :  $b \in K$  and  $bNu$ :  $\neg \text{vert-adj } b \ u$ 
      using  $bA$  by auto
      have  $bB$ :  $b \in \{x \in K. \neg \text{vert-adj } x \ v\}$ 
      using  $AB \ bA$  by blast
      have  $bNv$ :  $\neg \text{vert-adj } b \ v$ 
      using  $bB$  by auto
      have  $uv-ne$ :  $u \neq v$ 
      using vert-adj-neq[ $OF \ uv$ ] .
      have  $Kclq$ : is-clique  $K$ 
      using  $Kmax$  by (auto simp: maximum-clique-def cliques-def)
      have  $ab$ :  $\text{vert-adj } a \ b$ 
      using clique-pair-adj[ $OF \ Kclq \ aK \ bK \ ab-ne$ ] .
      have  $dist$ : distinct [ $a, b, u, v$ ]
      using  $aK \ bK \ uO \ vO \ ab-ne \ uv-ne$  by auto
      have induces-2K2  $a \ b \ u \ v$ 
      using  $dist \ ab \ uv \ aNu \ aNv \ bNu \ bNv$ 
      by (simp add: induces-2K2-def)
      then have has-induced-2K2
      by (auto simp: has-induced-2K2-def)
      with  $fh$  show False
      by (simp add: foldes-hammer-free-def)
    qed

```

```

qed
next
  show  $\{a\} \subseteq \{x \in K. \neg \text{vert-adj } x \ u\}$ 
    using aK aNu by auto
qed
show thesis
  using that Aeq aK aNu aNv by blast
qed

lemma orient-outside-edge-for-swap:
  assumes fh: foldes-hammer-free
    and Kmax: maximum-clique K
    and uO:  $u \in V - K$ 
    and vO:  $v \in V - K$ 
    and uv: vert-adj u v
  obtains u' v' a where
     $\{u', v'\} = \{u, v\}$ 
    and  $u' \in V - K$ 
    and  $v' \in V - K$ 
    and vert-adj u' v'
    and  $\{x \in K. \neg \text{vert-adj } x \ u'\} = \{a\}$ 
    and  $a \in K$ 
    and  $\neg \text{vert-adj } a \ u'$ 
    and  $\neg \text{vert-adj } a \ v'$ 
proof -
  have nested:
     $\{x \in K. \neg \text{vert-adj } x \ u\} \subseteq \{x \in K. \neg \text{vert-adj } x \ v\} \vee$ 
     $\{x \in K. \neg \text{vert-adj } x \ v\} \subseteq \{x \in K. \neg \text{vert-adj } x \ u\}$ 
    using outside-adjacent-nonneighbor-sets-nested[OF fh Kmax uO vO uv] .
  then show thesis
proof
  assume AB:  $\{x \in K. \neg \text{vert-adj } x \ u\} \subseteq \{x \in K. \neg \text{vert-adj } x \ v\}$ 
  obtain a where Aeq:  $\{x \in K. \neg \text{vert-adj } x \ u\} = \{a\}$  and aK:  $a \in K$ 
    and aNu:  $\neg \text{vert-adj } a \ u$  and aNv:  $\neg \text{vert-adj } a \ v$ 
    using outside-adjacent-smaller-nonneighbor-set-singleton[OF fh Kmax uO vO
uv AB] by blast
  show thesis
    using that[of u v a] uO vO uv Aeq aK aNu aNv by simp
next
  assume BA:  $\{x \in K. \neg \text{vert-adj } x \ v\} \subseteq \{x \in K. \neg \text{vert-adj } x \ u\}$ 
  have vu: vert-adj v u
    using uv by (simp add: vert-adj-sym)
  obtain a where Beq:  $\{x \in K. \neg \text{vert-adj } x \ v\} = \{a\}$  and aK:  $a \in K$ 
    and aNv:  $\neg \text{vert-adj } a \ v$  and aNu:  $\neg \text{vert-adj } a \ u$ 
    using outside-adjacent-smaller-nonneighbor-set-singleton[OF fh Kmax vO uO
vu BA] by blast
  show thesis
    using that[of v u a] vO uO vu Beq aK aNv aNu by auto
qed

```

qed

lemma *replace-unique-nonneighbor-maximum-clique:*

assumes *Kmax: maximum-clique K*

and *uO: $u \in V - K$*

and *Au: $\{x \in K. \neg \text{vert-adj } x \ u\} = \{a\}$*

shows *maximum-clique (insert u (K - {a}))*

proof -

have *Kclq: is-clique K*

using *Kmax* **by** (auto simp: maximum-clique-def cliques-def)

have *aK: $a \in K$ and aNu: $\neg \text{vert-adj } a \ u$*

using *Au* **by** auto

have *uV: $u \in V$ and unotK: $u \notin K$*

using *uO* **by** auto

have *Ksub: $K - \{a\} \subseteq K$*

by auto

have *Kminclq: is-clique (K - {a})*

using *clique-mono[OF Kclq Ksub]* .

have *uadj: $\forall x \in K - \{a\}. \text{vert-adj } x \ u$*

proof

fix *x*

assume *xK: $x \in K - \{a\}$*

have *xK0: $x \in K$ and xnea: $x \neq a$*

using *xK* **by** auto

have *$x \notin \{x \in K. \neg \text{vert-adj } x \ u\}$*

using *Au xnea* **by** auto

then show *vert-adj x u*

using *xK0* **by** auto

qed

have *newclq: is-clique (insert u (K - {a}))*

using *insert-clique-if-all-adj[OF Kminclq uV] unotK uadj* **by** auto

have *finK: finite K*

using *maximum-clique-finite[OF Kmax]* .

have *newcard: $\text{card (insert u (K - \{a\}))} = \text{card } K$*

proof -

have *u-not-Ka: $u \notin K - \{a\}$*

using *unotK* **by** auto

have *$\text{card (insert u (K - \{a\}))} = \text{Suc (card (K - \{a\}))}$*

using *finK u-not-Ka* **by** simp

also have *$\dots = \text{Suc (card } K - 1)$*

using *finK aK* **by** simp

also have *$\dots = \text{card } K$*

proof -

have *$K \neq \{\}$*

using *aK* **by** auto

have *$0 < \text{card } K$*

using *finK $\langle K \neq \{\} \rangle$* **by** (simp add: card-gt-0-iff)

then show *?thesis*

by arith

```

qed
finally show ?thesis .
qed
have newin: insert u (K - {a}) ∈ cliques
  using newclq by (simp add: cliques-def)
have newbound: ∀ C ∈ cliques. card C ≤ card (insert u (K - {a}))
proof
  fix C
  assume Cin: C ∈ cliques
  from Kmax have ∀ C ∈ cliques. card C ≤ card K
    by (simp add: maximum-clique-def)
  then show card C ≤ card (insert u (K - {a}))
    using Cin newcard by simp
qed
show ?thesis
  using newin newbound by (simp add: maximum-clique-def)
qed

lemma replacement-edge-transfer:
  assumes fh: foldes-hammer-free
    and Kmax: maximum-clique K
    and uO: u ∈ V - K
    and vO: v ∈ V - K
    and uv: vert-adj u v
    and Au: {x ∈ K. ¬ vert-adj x u} = {a}
    and aNv: ¬ vert-adj a v
    and zO: z ∈ V - K
    and zne: z ≠ u
    and az: vert-adj a z
  shows vert-adj u z
proof (rule ccontr)
  assume uz: ¬ vert-adj u z
  have aK: a ∈ K and aNu: ¬ vert-adj a u
    using Au by auto
  have uv-ne: u ≠ v
    using vert-adj-neq[OF uv] .
  have vz: vert-adj v z
proof (rule ccontr)
  assume vz-not: ¬ vert-adj v z
  have zv-ne: z ≠ v
    using az aNv by auto
  have dist: distinct [a, z, u, v]
    using aK uO vO zO uv-ne zne az zv-ne by auto
  have zNu: ¬ vert-adj z u
    using uz by (simp add: vert-adj-sym)
  have zNv: ¬ vert-adj z v
    using vz-not by (simp add: vert-adj-sym)
  have induces-2K2 a z u v
    using dist az uv aNu aNv zNu zNv

```

```

    by (simp add: induces-2K2-def)
  then have has-induced-2K2
    by (auto simp: has-induced-2K2-def)
  with fh show False
    by (simp add: foldes-hammer-free-def)
qed
have zv: vert-adj z v
  using vz by (simp add: vert-adj-sym)
have nested:
  {x∈K. ¬ vert-adj x z} ⊆ {x∈K. ¬ vert-adj x v} ∨
  {x∈K. ¬ vert-adj x v} ⊆ {x∈K. ¬ vert-adj x z}
  using outside-adjacent-nonneighbor-sets-nested[OF fh Kmax zO vO zv] .
have zV: z ∈ V and znotK: z ∉ K
  using zO by auto
obtain c where cK: c ∈ K and cNz: ¬ vert-adj c z
  using maximum-clique-outside-has-nonneighbor[OF Kmax zV znotK] by blast
have CsubB: {x∈K. ¬ vert-adj x z} ⊆ {x∈K. ¬ vert-adj x v}
proof -
  from nested show ?thesis
proof
  assume h: {x∈K. ¬ vert-adj x z} ⊆ {x∈K. ¬ vert-adj x v}
  show ?thesis by fact
next
  assume h: {x∈K. ¬ vert-adj x v} ⊆ {x∈K. ¬ vert-adj x z}
  have a ∈ {x∈K. ¬ vert-adj x z}
    using h aK aNv by auto
  then have aNz: ¬ vert-adj a z
    by auto
  then have False
    using az by contradiction
  then show ?thesis
    by blast
qed
qed
have cNv: ¬ vert-adj c v
  using CsubB cK cNz by auto
have ca-ne: c ≠ a
  using az cK cNz by auto
have cu: vert-adj c u
proof -
  have c ∉ {x∈K. ¬ vert-adj x u}
    using Au ca-ne by auto
  then show ?thesis
    using cK by auto
qed
have Kclq: is-clique K
  using Kmax by (auto simp: maximum-clique-def cliques-def)
have ac: vert-adj a c
  using clique-pair-adj[OF Kclq aK cK ca-ne[symmetric]] .

```

```

have zv: vert-adj z v
  using vz by (simp add: vert-adj-sym)
have vu: vert-adj v u
  using uv by (simp add: vert-adj-sym)
have uc: vert-adj u c
  using cu by (simp add: vert-adj-sym)
have ca: vert-adj c a
  using ac by (simp add: vert-adj-sym)
have zNu:  $\neg$  vert-adj z u
  using uz by (simp add: vert-adj-sym)
have zNc:  $\neg$  vert-adj z c
  using cNz by (simp add: vert-adj-sym)
have vNc:  $\neg$  vert-adj v c
  using cNv by (simp add: vert-adj-sym)
have dist: distinct [a, z, v, u, c]
  using aK cK uO vO zO uv-ne zne az ca-ne aNv by auto
have induces-C5 a z v u c
  using dist az zv vu uc ca aNv aNu zNu zNc vNc
  by (simp add: induces-C5-def)
then have has-induced-C5
  unfolding has-induced-C5-def by blast
with fh show False
  by (simp add: foldes-hammer-free-def)
qed

```

```

lemma outside-set-after-swap:
  assumes uO:  $u \in V - K$ 
  and aV:  $a \in V$ 
  and aK:  $a \in K$ 
  shows  $V - \text{insert } u (K - \{a\}) = \text{insert } a ((V - K) - \{u\})$ 
  using assms by auto

```

```

lemma swap-outside-edges-inj:
  assumes uO:  $u \in V - K$ 
  and aK:  $a \in K$ 
  shows inj-on (%e. if  $a \in e$  then  $\text{insert } u (e - \{a\})$  else  $e$ )
    (induced-edges (V - insert u (K - {a})))
proof (rule inj-onI)
  let ?I' = V - insert u (K - {a})
  let ?f = %e. if  $a \in e$  then  $\text{insert } u (e - \{a\})$  else  $e$ 
  fix e1 e2
  assume e1I:  $e1 \in \text{induced-edges } ?I'$ 
  and e2I:  $e2 \in \text{induced-edges } ?I'$ 
  and eq:  $?f e1 = ?f e2$ 
  have u-not-I':  $u \notin ?I'$ 
  using uO by auto
  have e1ss:  $e1 \subseteq ?I'$  and e2ss:  $e2 \subseteq ?I'$ 
  using e1I e2I by (auto simp: induced-edges-def)
  show e1 = e2

```

```

proof (cases a ∈ e1)
  case False
  then have f1: ?f e1 = e1
    by simp
  show ?thesis
proof (cases a ∈ e2)
  case False
  with eq f1 show ?thesis
    by simp
next
  case True
  then have u ∈ ?f e2
    by simp
  then have u ∈ e1
    using eq f1 by simp
  with e1ss u-not-I' show ?thesis
    by blast
qed
next
  case e1-has-a: True
  show ?thesis
proof (cases a ∈ e2)
  case False
  then have f2: ?f e2 = e2
    by simp
  have u ∈ ?f e1
    using e1-has-a by simp
  then have u ∈ e2
    using eq f2 by simp
  with e2ss u-not-I' show ?thesis
    by blast
next
  case e2-has-a: True
  have u-not-e1: u ∉ e1 - {a} and u-not-e2: u ∉ e2 - {a}
    using e1ss e2ss u-not-I' by blast+
  have insert u (e1 - {a}) = insert u (e2 - {a})
    using eq e1-has-a e2-has-a by simp
  then have diff-eq: e1 - {a} = e2 - {a}
  proof -
    have (insert u (e1 - {a})) - {u} = (insert u (e2 - {a})) - {u}
      using ⟨insert u (e1 - {a}) = insert u (e2 - {a})⟩ by simp
    then show ?thesis
      using u-not-e1 u-not-e2 by auto
  qed
  have e1 = insert a (e1 - {a})
    using e1-has-a by blast
  moreover have e2 = insert a (e2 - {a})
    using e2-has-a by blast
  ultimately show ?thesis

```

```

    using diff-eq by simp
  qed
qed
qed

lemma swap-outside-edges-image-subset:
  assumes fh: foldes-hammer-free
    and Kmax: maximum-clique K
    and uO:  $u \in V - K$ 
    and vO:  $v \in V - K$ 
    and uv: vert-adj u v
    and Au:  $\{x \in K. \neg \text{vert-adj } x \ u\} = \{a\}$ 
    and aNv:  $\neg \text{vert-adj } a \ v$ 
  shows (%e. if  $a \in e$  then insert u ( $e - \{a\}$ ) else e) '
    induced-edges (V - insert u (K - {a}))
     $\subseteq$  induced-edges (V - K)
proof
  let ?I = V - K
  let ?I' = V - insert u (K - {a})
  let ?f = %e. if  $a \in e$  then insert u ( $e - \{a\}$ ) else e
  have aK:  $a \in K$ 
    using Au by auto
  have aV:  $a \in V$ 
    using Kmax aK by (auto simp: maximum-clique-def cliques-def is-clique-def)
  have Ieq:  $?I' = \text{insert } a \ (?I - \{u\})$ 
    using outside-set-after-swap[OF uO aV aK] .
  fix x
  assume xin:  $x \in ?f \text{ ' induced-edges } ?I'$ 
  then obtain e where eI:  $e \in \text{induced-edges } ?I'$  and xeq:  $x = ?f \ e$ 
    by blast
  have ess:  $e \subseteq ?I'$  and eE:  $e \in E$ 
    using eI by (auto simp: induced-edges-def)
  show  $x \in \text{induced-edges } ?I$ 
proof (cases  $a \in e$ )
  case False
  then have xeqe:  $x = e$ 
    using xeq by simp
  have e  $\subseteq ?I$ 
  proof
    fix y
    assume y  $\in e$ 
    with ess False show  $y \in ?I$ 
      by (auto simp: Ieq)
  qed
  with eE xeqe show ?thesis
    by (auto simp: induced-edges-def)
next
  case True
  from eE have ecard:  $\text{card } e = 2$ 

```

```

    using two-edges by auto
  from True ecard obtain z where e: e = {a, z} and zne: z ≠ a
  by (auto simp: card-2-iff)
  have zI: z ∈ ?I - {u}
  using ess True e zne by (auto simp: Ieq)
  then have zO: z ∈ V - K and zne-u: z ≠ u
  by auto
  have az: vert-adj a z
  using eE e by (simp add: vert-adj-def)
  have uz: vert-adj u z
  using replacement-edge-transfer[OF fh Kmax uO vO uv Au aNv zO zne-u az]
.
  have xeqz: x = {u, z}
  using xeq True e zne by simp
  have uzE: {u, z} ∈ E
  using uz by (simp add: vert-adj-def)
  have uz-sub: {u, z} ⊆ ?I
  using uO zO by auto
  show ?thesis
  using xeqz uzE uz-sub by (auto simp: induced-edges-def)
qed
qed

lemma swap-outside-edges-missing-uv:
  assumes uO: u ∈ V - K
  and vO: v ∈ V - K
  and uv: vert-adj u v
  and aK: a ∈ K
  and aNv: ¬ vert-adj a v
  shows {u, v} ∈ induced-edges (V - K)
  and {u, v} ∉ (%e. if a ∈ e then insert u (e - {a}) else e) '
    induced-edges (V - insert u (K - {a}))
  proof -
    have uv-in: {u, v} ∈ induced-edges (V - K)
    using uO vO uv by (auto simp: induced-edges-def vert-adj-def)
    show {u, v} ∈ induced-edges (V - K)
    using uv-in .
    show {u, v} ∉ (%e. if a ∈ e then insert u (e - {a}) else e) '
      induced-edges (V - insert u (K - {a}))
    proof
      let ?I' = V - insert u (K - {a})
      let ?f = %e. if a ∈ e then insert u (e - {a}) else e
      assume img: {u, v} ∈ ?f ' induced-edges ?I'
      then obtain e where eI: e ∈ induced-edges ?I' and eq: {u, v} = ?f e
      by blast
      have u-not-I': u ∉ ?I'
      using uO by auto
      have ess: e ⊆ ?I' and eE: e ∈ E
      using eI by (auto simp: induced-edges-def)

```

```

have uv-ne:  $u \neq v$ 
  using vert-adj-neq[OF uv] .
show False
proof (cases  $a \in e$ )
  case False
  then have  $\{u, v\} = e$ 
    using eq by simp
  then have  $u \in e$ 
    by auto
  with ess u-not-I' show False
    by blast
next
case True
from eE have ecard:  $\text{card } e = 2$ 
  using two-edges by auto
from True ecard obtain z where e:  $e = \{a, z\}$  and zne:  $z \neq a$ 
  by (auto simp: card-2-iff)
have zI':  $z \in ?I'$ 
  using ess True e zne by auto
then have zne-u:  $z \neq u$ 
  using u-not-I' by auto
have  $\{u, v\} = \{u, z\}$ 
  using eq True e zne by simp
then have zeqv:  $z = v$ 
  using uv-ne zne-u by auto
have vert-adj a z
  using eE e by (simp add: vert-adj-def)
with aNv zeqv show False
  by simp
qed
qed
qed

lemma outside-edge-count-swap-strict:
  assumes fh: foldes-hammer-free
  and Kmax: maximum-clique K
  and uO:  $u \in V - K$ 
  and vO:  $v \in V - K$ 
  and uv:  $\text{vert-adj } u \ v$ 
  and Au:  $\{x \in K. \neg \text{vert-adj } x \ u\} = \{a\}$ 
  and aNv:  $\neg \text{vert-adj } a \ v$ 
  shows  $\text{outside-edge-count } (\text{insert } u \ (K - \{a\})) < \text{outside-edge-count } K$ 
proof -
  let ?I =  $V - K$ 
  let ?I' =  $V - \text{insert } u \ (K - \{a\})$ 
  let ?f =  $\%e. \text{ if } a \in e \text{ then insert } u \ (e - \{a\}) \text{ else } e$ 
  have aK:  $a \in K$ 
    using Au by auto
  have inj: inj-on ?f (induced-edges ?I')

```

```

    using swap-outside-edges-inj[OF uO aK] .
  have imgsub: ?f ' induced-edges ?I'  $\subseteq$  induced-edges ?I
    using swap-outside-edges-image-subset[OF fh Kmax uO vO uv Au aNv] .
  have uv-in: {u, v}  $\in$  induced-edges ?I
    using swap-outside-edges-missing-uv[OF uO vO uv aK aNv] by blast
  have uv-not-img: {u, v}  $\notin$  ?f ' induced-edges ?I'
    using swap-outside-edges-missing-uv[OF uO vO uv aK aNv] by blast
  have psub: ?f ' induced-edges ?I'  $\subset$  induced-edges ?I
    using imgsub uv-in uv-not-img by blast
  have finI': finite (induced-edges ?I')
    by (rule finite-induced-edges-subset-V) auto
  have finI: finite (induced-edges ?I)
    by (rule finite-induced-edges-subset-V) auto
  have card-img: card (?f ' induced-edges ?I') = card (induced-edges ?I')
    by (rule card-image[OF inj])
  have card (induced-edges ?I') = card (?f ' induced-edges ?I')
    using card-img by simp
  also have ... < card (induced-edges ?I)
    using finI psub by (rule psubset-card-mono)
  finally show ?thesis
    by (simp add: outside-edge-count-def)
qed

```

theorem *foldes-hammer-converse:*

assumes fh: *foldes-hammer-free*

shows *is-split-graph*

proof –

obtain K **where** Kmax: *maximum-clique* K

and Kbest: $\forall L.$ *maximum-clique* L $\longrightarrow$ *outside-edge-count* K $\leq$ *outside-edge-count* L

using *obtain-best-clique* **by** blast

have Kclq: *is-clique* K

using Kmax **by** (auto simp: *maximum-clique-def* *cliques-def*)

have indep: *is-independent-set* (V – K)

proof (rule ccontr)

assume not-indep: $\neg$ *is-independent-set* (V – K)

then obtain u v **where** uO: $u \in V - K$ **and** vO: $v \in V - K$ **and** uv: *vert-adj*

u v

by (auto simp: *is-independent-alt*)

obtain u' v' a **where** uvset: {u', v'} = {u, v}

and u'O: $u' \in V - K$ **and** v'O: $v' \in V - K$ **and** uv': *vert-adj* u' v'

and Au': {x $\in$ K. $\neg$ *vert-adj* x u'} = {a}

and aK: $a \in K$ **and** aNu': $\neg$ *vert-adj* a u' **and** aNv': $\neg$ *vert-adj* a v'

using *orient-outside-edge-for-swap*[OF fh Kmax uO vO uv] **by** blast

let ?K' = *insert* u' (K – {a})

have K'max: *maximum-clique* ?K'

using *replace-unique-nonneighbor-maximum-clique*[OF Kmax u'O Au] .

have dec: *outside-edge-count* ?K' < *outside-edge-count* K

using *outside-edge-count-swap-strict*[OF fh Kmax u'O v'O uv' Au' aNv'] .

```

    have outside-edge-count  $K \leq$  outside-edge-count ? $K'$ 
      using  $K_{\text{best}} K'_{\text{max}}$  by blast
    with dec show False
      by simp
  qed
  show ?thesis
    using  $K_{\text{clq}} \text{ indep}$  by (auto simp: split-graph-alt)
qed

theorem foldes-hammer:
  is-split-graph  $\longleftrightarrow$  foldes-hammer-free
  using foldes-hammer-converse split-graph-imp-foldes-hammer-free by blast

end

end

```

References

- [1] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: A Survey*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999. DOI: <https://doi.org/10.1137/1.9780898719796>.
- [2] S. Földes and P. L. Hammer. Split graphs. *Congressus Numerantium*, 19:311–315, 1977.