Matroids

Jonas Keinholz

October 13, 2025

Abstract

This article defines combinatorial structures known as *Independence Systems* and *Matroids* and provides basic concepts and theorems related to them. These structures play an important role in combinatorial optimisation, e. g. greedy algorithms such as Kruskal's algorithm. The development is based on Oxley's 'What is a Matroid?' [1].

Contents

1 Independence systems **3**

1.1 Sub-independence systems . . . 3

1.2 Bases . . . 5

1.3 Circuits . . . 8

1.4 Relation between independence and bases . . . 12

1.5 Relation between dependence and circuits . . . 14

1.6 Ranks . . . 15

2 Matroids **17**

2.1 Minors . . . 18

2.2 Bases . . . 19

2.3 Circuits . . . 20

2.4 Ranks . . . 26

2.5 Closure . . . 33

1 Independence systems

theory *Indep-System*
  imports *Main*
begin

lemma *finite-psubset-inc-induct*:
  assumes *finite* A $X \subseteq A$
  assumes $\bigwedge X. (\bigwedge Y. X \subset Y \Longrightarrow Y \subseteq A \Longrightarrow P\ Y) \Longrightarrow P\ X$
  shows $P\ X$
proof −
  have *wf*: *wf* $\{(X,Y).\ Y \subset X \wedge X \subseteq A\}$
    by (*rule wf-bounded-set*[**where** $ub = \lambda$-. A **and** $f = id$]) (*auto simp add*: ‹*finite* A›)
  show *?thesis*
  proof (*induction X rule*: *wf-induct*[*OF wf*, *case-names step*])
    case (*step X*)
    then show *?case* **using** *assms*(*3*)[*of X*] **by** *blast*
  qed
qed

An *independence system* consists of a finite ground set together with an independence predicate over the sets of this ground set. At least one set of the carrier is independent and subsets of independent sets are also independent.

locale *indep-system* =
  fixes *carrier* :: *'a set*
  fixes *indep* :: *'a set* $\Rightarrow$ *bool*
  assumes *carrier-finite*: *finite carrier*
  assumes *indep-subset-carrier*: *indep* $X \Longrightarrow X \subseteq$ *carrier*
  assumes *indep-ex*: $\exists X.$ *indep* X
  assumes *indep-subset*: *indep* $X \Longrightarrow Y \subseteq X \Longrightarrow$ *indep* Y
begin

lemmas *psubset-inc-induct* [*case-names carrier step*] = *finite-psubset-inc-induct*[*OF carrier-finite*]
lemmas *indep-finite* [*simp*] = *finite-subset*[*OF indep-subset-carrier carrier-finite*]

The empty set is independent.

lemma *indep-empty* [*simp*]: *indep* {}
  using *indep-ex indep-subset* **by** *auto*

1.1 Sub-independence systems

A subset of the ground set induces an independence system.

definition *indep-in* **where** *indep-in* $\mathcal{E}$ $X \longleftrightarrow X \subseteq \mathcal{E} \wedge$ *indep* X

lemma *indep-inI*:
  assumes $X \subseteq \mathcal{E}$

assumes *indep X*
shows *indep-in* $\mathcal{E}$ *X*
using *assms* **unfolding** *indep-in-def* **by** *auto*

lemma *indep-in-subI*: *indep-in* $\mathcal{E}$ $X \Longrightarrow$ *indep-in* $\mathcal{E}'$ $(X \cap \mathcal{E}')$
using *indep-subset* **unfolding** *indep-in-def* **by** *auto*

lemma *dep-in-subI*:
assumes $X \subseteq \mathcal{E}'$
shows $\neg$ *indep-in* $\mathcal{E}'$ $X \Longrightarrow \neg$ *indep-in* $\mathcal{E}$ *X*
using *assms* **unfolding** *indep-in-def* **by** *auto*

lemma *indep-in-subset-carrier*: *indep-in* $\mathcal{E}$ $X \Longrightarrow X \subseteq \mathcal{E}$
unfolding *indep-in-def* **by** *auto*

lemma *indep-in-subI-subset*:
assumes $\mathcal{E}' \subseteq \mathcal{E}$
assumes *indep-in* $\mathcal{E}'$ *X*
shows *indep-in* $\mathcal{E}$ *X*
proof −
have *indep-in* $\mathcal{E}$ $(X \cap \mathcal{E})$ **using** *assms indep-in-subI* **by** *auto*
moreover have $X \cap \mathcal{E} = X$ **using** *assms indep-in-subset-carrier* **by** *auto*
ultimately show *?thesis* **by** *auto*
qed

lemma *indep-in-supI*:
assumes $X \subseteq \mathcal{E}'$ $\mathcal{E}' \subseteq \mathcal{E}$
assumes *indep-in* $\mathcal{E}$ *X*
shows *indep-in* $\mathcal{E}'$ *X*
proof −
have $X \cap \mathcal{E}' = X$ **using** *assms* **by** *auto*
then show *?thesis* **using** *assms indep-in-subI*[**where** $\mathcal{E} = \mathcal{E}$ **and** $\mathcal{E}' = \mathcal{E}'$ **and** $X = X$] **by** *auto*
qed

lemma *indep-in-indep*: *indep-in* $\mathcal{E}$ $X \Longrightarrow$ *indep X*
unfolding *indep-in-def* **by** *auto*

lemmas *indep-inD* = *indep-in-subset-carrier indep-in-indep*

lemma *indep-system-subset* [*simp*, *intro*]:
assumes $\mathcal{E} \subseteq$ *carrier*
shows *indep-system* $\mathcal{E}$ (*indep-in* $\mathcal{E}$)
unfolding *indep-system-def indep-in-def*
using *finite-subset*[*OF assms carrier-finite*] *indep-subset* **by** *auto*

We will work a lot with different sub structures. Therefore, every definition 'foo' will have a counterpart 'foo_in' which has the ground set as an additional parameter. Furthermore, every result about 'foo' will have another result about 'foo_in'. With this, we usually don't have to work with **interpretation** in proofs.

context
  fixes $\mathcal{E}$
  assumes $\mathcal{E} \subseteq$ *carrier*
begin

interpretation $\mathcal{E}$: *indep-system* $\mathcal{E}$ *indep-in* $\mathcal{E}$
  using ‹$\mathcal{E} \subseteq$ *carrier*› **by** *auto*

lemma *indep-in-sub-cong*:
  assumes $\mathcal{E}' \subseteq \mathcal{E}$
  shows $\mathcal{E}$.*indep-in* $\mathcal{E}'$ $X \longleftrightarrow$ *indep-in* $\mathcal{E}'$ X
  unfolding $\mathcal{E}$.*indep-in-def* *indep-in-def* **using** *assms* **by** *auto*

lemmas *indep-in-ex* = $\mathcal{E}$.*indep-ex*
lemmas *indep-in-subset* = $\mathcal{E}$.*indep-subset*
lemmas *indep-in-empty* = $\mathcal{E}$.*indep-empty*

end

1.2 Bases

A *basis* is a maximal independent set, i. e. an independent set which becomes dependent on inserting any element of the ground set.

definition *basis* **where** *basis* $X \longleftrightarrow$ *indep* $X \wedge (\forall x \in$ *carrier* $- X.\ \neg$ *indep* (*insert* x X))

lemma *basisI*:
  assumes *indep* X
  assumes $\bigwedge x.\ x \in$ *carrier* $- X \Longrightarrow \neg$ *indep* (*insert* x X)
  shows *basis* X
  using *assms* **unfolding** *basis-def* **by** *auto*

lemma *basis-indep*: *basis* $X \Longrightarrow$ *indep* X
  unfolding *basis-def* **by** *auto*

lemma *basis-max-indep*: *basis* $X \Longrightarrow x \in$ *carrier* $- X \Longrightarrow \neg$ *indep* (*insert* x X)
  unfolding *basis-def* **by** *auto*

lemmas *basisD* = *basis-indep basis-max-indep*
lemmas *basis-subset-carrier* = *indep-subset-carrier*[*OF basis-indep*]
lemmas *basis-finite* [*simp*] = *indep-finite*[*OF basis-indep*]

lemma *indep-not-basis*:
  assumes *indep* X
  assumes $\neg$ *basis* X
  shows $\exists x \in$ *carrier* $- X.$ *indep* (*insert* x X)

using *assms basisI* **by** *auto*

lemma *basis-subset-eq*:
assumes *basis* B_1
assumes *basis* B_2
assumes $B_1 \subseteq B_2$
shows $B_1 = B_2$
proof (*rule ccontr*)
assume $B_1 \neq B_2$
then obtain x **where** x: $x \in B_2 - B_1$ **using** *assms* **by** *auto*
then have *insert* x $B_1 \subseteq B_2$ **using** *assms* **by** *auto*
then have *indep* (*insert* x B_1) **using** *assms basis-indep*[*of* B_2] *indep-subset* **by** *auto*
moreover have $x \in$ *carrier* $- B_1$ **using** *assms x basis-subset-carrier* **by** *auto*
ultimately show *False* **using** *assms basisD* **by** *auto*
qed

definition *basis-in* **where**
basis-in $\mathcal{E}$ $X \longleftrightarrow$ *indep-system.basis* $\mathcal{E}$ (*indep-in* $\mathcal{E}$) X

lemma *basis-iff-basis-in*: *basis* $B \longleftrightarrow$ *basis-in carrier* B
proof −
interpret $\mathcal{E}$: *indep-system carrier indep-in carrier*
by *auto*

show *basis* $B \longleftrightarrow$ *basis-in carrier* B
unfolding *basis-in-def*
proof (*standard*, *goal-cases LTR RTL*)
case *LTR*
show *?case*
proof (*rule* $\mathcal{E}$.*basisI*)
show *indep-in carrier* B **using** *LTR basisD indep-subset-carrier indep-inI* **by** *auto*
next
fix x
assume $x \in$ *carrier* $- B$
then have $\neg$ *indep* (*insert* x B) **using** *LTR basisD* **by** *auto*
then show $\neg$ *indep-in carrier* (*insert* x B) **using** *indep-inD* **by** *auto*
qed
next
case *RTL*
show *?case*
proof (*rule basisI*)
show *indep* B **using** *RTL* $\mathcal{E}$.*basis-indep indep-inD* **by** *blast*
next
fix x
assume $x \in$ *carrier* $- B$
then have $\neg$ *indep-in carrier* (*insert* x B) **using** *RTL* $\mathcal{E}$.*basisD* **by** *auto*
then show $\neg$ *indep* (*insert* x B) **using** *indep-subset-carrier indep-inI* **by** *blast*

```
    qed
  qed
qed

context
  fixes ℰ
  assumes ℰ ⊆ carrier
begin

interpretation ℰ: indep-system ℰ indep-in ℰ
  using ‹ℰ ⊆ carrier› by auto

lemma basis-inI-aux: ℰ.basis X ⟹ basis-in ℰ X
  unfolding basis-in-def by auto

lemma basis-inD-aux: basis-in ℰ X ⟹ ℰ.basis X
  unfolding basis-in-def by auto

lemma not-basis-inD-aux: ¬ basis-in ℰ X ⟹ ¬ ℰ.basis X
  using basis-inI-aux by auto

lemmas basis-inI = basis-inI-aux[OF ℰ.basisI]
lemmas basis-in-indep-in = ℰ.basis-indep[OF basis-inD-aux]
lemmas basis-in-max-indep-in = ℰ.basis-max-indep[OF basis-inD-aux]
lemmas basis-inD = ℰ.basisD[OF basis-inD-aux]
lemmas basis-in-subset-carrier = ℰ.basis-subset-carrier[OF basis-inD-aux]
lemmas basis-in-finite = ℰ.basis-finite[OF basis-inD-aux]
lemmas indep-in-not-basis-in = ℰ.indep-not-basis[OF - not-basis-inD-aux]
lemmas basis-in-subset-eq = ℰ.basis-subset-eq[OF basis-inD-aux basis-inD-aux]

end

context
  fixes ℰ
  assumes *: ℰ ⊆ carrier
begin

interpretation ℰ: indep-system ℰ indep-in ℰ
  using * by auto

lemma basis-in-sub-cong:
  assumes ℰ' ⊆ ℰ
  shows ℰ.basis-in ℰ' B ⟷ basis-in ℰ' B
proof (safe, goal-cases LTR RTL)
  case LTR
  show ?case
  proof (rule basis-inI)
    show ℰ' ⊆ carrier using assms * by auto
  next
```

show *indep-in* $\mathcal{E}'$ *B*
using $*$ *assms LTR* $\mathcal{E}$*.basis-in-subset-carrier* $\mathcal{E}$*.basis-in-indep-in indep-in-sub-cong*
by *auto*
next
fix *x*
assume $x \in \mathcal{E}' - B$
then show $\neg$ *indep-in* $\mathcal{E}'$ (*insert x B*)
using $*$ *assms LTR* $\mathcal{E}$*.basis-in-max-indep-in* $\mathcal{E}$*.basis-in-subset-carrier indep-in-sub-cong* **by** *auto*
qed
next
case *RTL*
show *?case*
proof (*rule* $\mathcal{E}$*.basis-inI*)
show $\mathcal{E}' \subseteq \mathcal{E}$ **using** *assms* **by** *auto*
next
show $\mathcal{E}$*.indep-in* $\mathcal{E}'$ *B*
using $*$ *assms RTL basis-in-subset-carrier basis-in-indep-in indep-in-sub-cong*
by *auto*
next
fix *x*
assume $x \in \mathcal{E}' - B$
then show $\neg$ $\mathcal{E}$*.indep-in* $\mathcal{E}'$ (*insert x B*)
using $*$ *assms RTL basis-in-max-indep-in basis-in-subset-carrier indep-in-sub-cong*
by *auto*
qed
qed

end

1.3 Circuits

A *circuit* is a minimal dependent set, i. e. a set which becomes independent on removing any element of the ground set.

definition *circuit* **where** *circuit* $X \longleftrightarrow X \subseteq$ *carrier* $\wedge \neg$ *indep* $X \wedge (\forall x \in X.$ *indep* $(X - \{x\}))$

lemma *circuitI*:
assumes $X \subseteq$ *carrier*
assumes $\neg$ *indep X*
assumes $\bigwedge x.\ x \in X \Longrightarrow$ *indep* $(X - \{x\})$
shows *circuit X*
using *assms* **unfolding** *circuit-def* **by** *auto*

lemma *circuit-subset-carrier*: *circuit* $X \Longrightarrow X \subseteq$ *carrier*
unfolding *circuit-def* **by** *auto*
lemmas *circuit-finite* [*simp*] = *finite-subset*[*OF circuit-subset-carrier carrier-finite*]

lemma *circuit-dep*: *circuit* $X \Longrightarrow \neg$ *indep X*

unfolding *circuit-def* **by** *auto*

lemma *circuit-min-dep*: $circuit\ X \Longrightarrow x \in X \Longrightarrow indep\ (X - \{x\})$
unfolding *circuit-def* **by** *auto*

lemmas *circuitD* = *circuit-subset-carrier circuit-dep circuit-min-dep*

lemma *circuit-nonempty*: $circuit\ X \Longrightarrow X \neq \{\}$
using *circuit-dep indep-empty* **by** *blast*

lemma *dep-not-circuit*:
assumes $X \subseteq carrier$
assumes $\neg\ indep\ X$
assumes $\neg\ circuit\ X$
shows $\exists\, x \in X.\ \neg\ indep\ (X - \{x\})$
using *assms circuitI* **by** *auto*

lemma *circuit-subset-eq*:
assumes *circuit* C_1
assumes *circuit* C_2
assumes $C_1 \subseteq C_2$
shows $C_1 = C_2$
proof (*rule ccontr*)
assume $C_1 \neq C_2$
then obtain x **where** $x \notin C_1$ $x \in C_2$ **using** *assms* **by** *auto*
then have *indep* C_1 **using** *indep-subset* ‹$C_1 \subseteq C_2$› *circuit-min-dep*[*OF* ‹*circuit* C_2›, *of x*] **by** *auto*
then show *False* **using** *assms circuitD* **by** *auto*
qed

definition *circuit-in* **where**
$circuit\text{-}in\ \mathcal{E}\ X \longleftrightarrow indep\text{-}system.circuit\ \mathcal{E}\ (indep\text{-}in\ \mathcal{E})\ X$

context
fixes $\mathcal{E}$
assumes $\mathcal{E} \subseteq carrier$
begin

interpretation $\mathcal{E}$: *indep-system* $\mathcal{E}$ *indep-in* $\mathcal{E}$
using ‹$\mathcal{E} \subseteq carrier$› **by** *auto*

lemma *circuit-inI-aux*: $\mathcal{E}.circuit\ X \Longrightarrow circuit\text{-}in\ \mathcal{E}\ X$
unfolding *circuit-in-def* **by** *auto*

lemma *circuit-inD-aux*: $circuit\text{-}in\ \mathcal{E}\ X \Longrightarrow \mathcal{E}.circuit\ X$
unfolding *circuit-in-def* **by** *auto*

lemma *not-circuit-inD-aux*: $\neg\ circuit\text{-}in\ \mathcal{E}\ X \Longrightarrow \neg\ \mathcal{E}.circuit\ X$
using *circuit-inI-aux* **by** *auto*

lemmas *circuit-inI* = *circuit-inI-aux*[*OF* $\mathcal{E}$.*circuitI*]

lemmas *circuit-in-subset-carrier* = $\mathcal{E}$.*circuit-subset-carrier*[*OF circuit-inD-aux*]
lemmas *circuit-in-finite* = $\mathcal{E}$.*circuit-finite*[*OF circuit-inD-aux*]
lemmas *circuit-in-dep-in* = $\mathcal{E}$.*circuit-dep*[*OF circuit-inD-aux*]
lemmas *circuit-in-min-dep-in* = $\mathcal{E}$.*circuit-min-dep*[*OF circuit-inD-aux*]
lemmas *circuit-inD* = $\mathcal{E}$.*circuitD*[*OF circuit-inD-aux*]
lemmas *circuit-in-nonempty* = $\mathcal{E}$.*circuit-nonempty*[*OF circuit-inD-aux*]
lemmas *dep-in-not-circuit-in* = $\mathcal{E}$.*dep-not-circuit*[*OF - - not-circuit-inD-aux*]
lemmas *circuit-in-subset-eq* = $\mathcal{E}$.*circuit-subset-eq*[*OF circuit-inD-aux circuit-inD-aux*]

end

lemma *circuit-in-subI*:
  assumes $\mathcal{E}' \subseteq \mathcal{E}$ $\mathcal{E} \subseteq$ *carrier*
  assumes *circuit-in* $\mathcal{E}'$ C
  shows *circuit-in* $\mathcal{E}$ C
proof (*rule circuit-inI*)
  show $\mathcal{E} \subseteq$ *carrier* **using** *assms* **by** *auto*
next
  show $C \subseteq \mathcal{E}$ **using** *assms circuit-in-subset-carrier*[*of* $\mathcal{E}'$ C] **by** *auto*
next
  show $\neg$ *indep-in* $\mathcal{E}$ C
    using *assms*
      circuit-in-dep-in[**where** $\mathcal{E} = \mathcal{E}'$ **and** $X = C$]
      circuit-in-subset-carrier dep-in-subI[**where** $\mathcal{E}' = \mathcal{E}'$ **and** $\mathcal{E} = \mathcal{E}$]
    by *auto*
next
  fix x
  assume $x \in C$
  then show *indep-in* $\mathcal{E}$ $(C - \{x\})$
    using *assms circuit-in-min-dep-in indep-in-subI-subset* **by** *auto*
qed

lemma *circuit-in-supI*:
  assumes $\mathcal{E}' \subseteq \mathcal{E}$ $\mathcal{E} \subseteq$ *carrier* $C \subseteq \mathcal{E}'$
  assumes *circuit-in* $\mathcal{E}$ C
  shows *circuit-in* $\mathcal{E}'$ C
proof (*rule circuit-inI*)
  show $\mathcal{E}' \subseteq$ *carrier* **using** *assms* **by** *auto*
next
  show $C \subseteq \mathcal{E}'$ **using** *assms* **by** *auto*
next
  have $\neg$ *indep-in* $\mathcal{E}$ C **using** *assms circuit-in-dep-in* **by** *auto*
  then show $\neg$ *indep-in* $\mathcal{E}'$ C **using** *assms dep-in-subI*[*of* C $\mathcal{E}$] **by** *auto*
next
  fix x
  assume $x \in C$

then have *indep-in* $\mathcal{E}$ $(C - \{x\})$ **using** *assms circuit-in-min-dep-in* **by** *auto*
then have *indep-in* $\mathcal{E}'$ $((C - \{x\}) \cap \mathcal{E}')$ **using** *indep-in-subI* **by** *auto*
moreover have $(C - \{x\}) \cap \mathcal{E}' = C - \{x\}$ **using** *assms circuit-in-subset-carrier* **by** *auto*
ultimately show *indep-in* $\mathcal{E}'$ $(C - \{x\})$ **by** *auto*
qed

context
fixes $\mathcal{E}$
assumes $*$: $\mathcal{E} \subseteq$ *carrier*
begin

interpretation $\mathcal{E}$: *indep-system* $\mathcal{E}$ *indep-in* $\mathcal{E}$
using $*$ **by** *auto*

lemma *circuit-in-sub-cong*:
assumes $\mathcal{E}' \subseteq \mathcal{E}$
shows $\mathcal{E}$*.circuit-in* $\mathcal{E}'$ $C \longleftrightarrow$ *circuit-in* $\mathcal{E}'$ C
proof (*safe*, *goal-cases LTR RTL*)
case *LTR*
show *?case*
proof (*rule circuit-inI*)
show $\mathcal{E}' \subseteq$ *carrier* **using** *assms* $*$ **by** *auto*
next
show $C \subseteq \mathcal{E}'$
using *assms LTR* $\mathcal{E}$*.circuit-in-subset-carrier* **by** *auto*
next
show $\neg$ *indep-in* $\mathcal{E}'$ C
using *assms LTR* $\mathcal{E}$*.circuit-in-dep-in indep-in-sub-cong*[*OF* $*$] **by** *auto*
next
fix x
assume $x \in C$
then show *indep-in* $\mathcal{E}'$ $(C - \{x\})$
using *assms LTR* $\mathcal{E}$*.circuit-in-min-dep-in indep-in-sub-cong*[*OF* $*$] **by** *auto*
qed
next
case *RTL*
show *?case*
proof (*rule* $\mathcal{E}$*.circuit-inI*)
show $\mathcal{E}' \subseteq \mathcal{E}$ **using** *assms* $*$ **by** *auto*
next
show $C \subseteq \mathcal{E}'$
using *assms* $*$ *RTL circuit-in-subset-carrier* **by** *auto*
next
show $\neg$ $\mathcal{E}$*.indep-in* $\mathcal{E}'$ C
using *assms* $*$ *RTL circuit-in-dep-in indep-in-sub-cong*[*OF* $*$] **by** *auto*
next
fix x
assume $x \in C$

then show $\mathcal{E}$*.indep-in* $\mathcal{E}'$ $(C - \{x\})$
  using *assms* $*$ *RTL circuit-in-min-dep-in indep-in-sub-cong*[*OF* $*$] **by** *auto*
 qed
qed

end

lemma *circuit-imp-circuit-in*:
 assumes *circuit C*
 shows *circuit-in carrier C*
proof (*rule circuit-inI*)
 show $C \subseteq$ *carrier* **using** *circuit-subset-carrier*[*OF assms*] **.**
next
 show $\neg$ *indep-in carrier C* **using** *circuit-dep*[*OF assms*] *indep-in-indep* **by** *auto*
next
 fix x
 assume $x \in C$
 then have *indep* $(C - \{x\})$ **using** *circuit-min-dep*[*OF assms*] **by** *auto*
 then show *indep-in carrier* $(C - \{x\})$ **using** *circuit-subset-carrier*[*OF assms*]
by (*auto intro*: *indep-inI*)
qed *auto*

1.4 Relation between independence and bases

A set is independent iff it is a subset of a basis.

lemma *indep-imp-subset-basis*:
 assumes *indep X*
 shows $\exists B.$ *basis* $B \wedge X \subseteq B$
 using *assms*
proof (*induction X rule*: *psubset-inc-induct*)
 case *carrier*
 show *?case* **using** *indep-subset-carrier*[*OF assms*] **.**
next
 case (*step X*)
 {
  assume $\neg$ *basis X*
  then obtain x **where** $x \in$ *carrier* $x \notin X$ *indep* (*insert* x X)
   using *step.prems indep-not-basis* **by** *auto*
  then have *?case* **using** *step.IH*[*of insert x X*] *indep-subset-carrier* **by** *auto*
 }
 then show *?case* **by** *auto*
qed

lemmas *subset-basis-imp-indep* = *indep-subset*[*OF basis-indep*]

lemma *indep-iff-subset-basis*: *indep X* $\longleftrightarrow$ $(\exists B.$ *basis* $B \wedge X \subseteq B)$
 using *indep-imp-subset-basis subset-basis-imp-indep* **by** *auto*

lemma *basis-ex*: $\exists B.$ *basis B*

using *indep-imp-subset-basis*[*OF indep-empty*] **by** *auto*

context
fixes $\mathcal{E}$
assumes $*$: $\mathcal{E} \subseteq$ *carrier*
begin

interpretation $\mathcal{E}$: *indep-system* $\mathcal{E}$ *indep-in* $\mathcal{E}$
using $*$ **by** *auto*

lemma *indep-in-imp-subset-basis-in*:
assumes *indep-in* $\mathcal{E}$ X
shows $\exists B.$ *basis-in* $\mathcal{E}$ $B \wedge X \subseteq B$
unfolding *basis-in-def* **using** $\mathcal{E}$.*indep-imp-subset-basis*[*OF assms*] **.**

lemmas *subset-basis-in-imp-indep-in* = *indep-in-subset*[*OF* $*$ *basis-in-indep-in*[*OF* $*$]]

lemma *indep-in-iff-subset-basis-in*: *indep-in* $\mathcal{E}$ $X \longleftrightarrow (\exists B.$ *basis-in* $\mathcal{E}$ $B \wedge X \subseteq B)$
using *indep-in-imp-subset-basis-in subset-basis-in-imp-indep-in* **by** *auto*

lemma *basis-in-ex*: $\exists B.$ *basis-in* $\mathcal{E}$ B
unfolding *basis-in-def* **using** $\mathcal{E}$.*basis-ex* **.**

lemma *basis-in-subI*:
assumes $\mathcal{E}' \subseteq \mathcal{E}$ $\mathcal{E} \subseteq$ *carrier*
assumes *basis-in* $\mathcal{E}'$ B
shows $\exists B' \subseteq \mathcal{E} - \mathcal{E}'.$ *basis-in* $\mathcal{E}$ $(B \cup B')$
proof $-$
have *indep-in* $\mathcal{E}$ B **using** *assms basis-in-indep-in indep-in-subI-subset* **by** *auto*
then obtain B' **where** B': *basis-in* $\mathcal{E}$ B' $B \subseteq B'$
using *assms indep-in-imp-subset-basis-in*[*of B*] **by** *auto*
show *?thesis*
proof (*rule exI*)
have $B' - B \subseteq \mathcal{E} - \mathcal{E}'$
proof
fix x
assume $*$: $x \in B' - B$
then have $x \in \mathcal{E}$ $x \notin B$
using *assms* ‹*basis-in* $\mathcal{E}$ B'› *basis-in-subset-carrier*[*of* $\mathcal{E}$] **by** *auto*
moreover {
assume $x \in \mathcal{E}'$
moreover have *indep-in* $\mathcal{E}$ (*insert* x B)
using $*$ *assms indep-in-subset*[*OF - basis-in-indep-in*] B' **by** *auto*
ultimately have *indep-in* $\mathcal{E}'$ (*insert* x B)
using *assms basis-in-subset-carrier* **unfolding** *indep-in-def* **by** *auto*
then have *False* **using** *assms* $*$ ‹$x \in \mathcal{E}'$› *basis-in-max-indep-in* **by** *auto*
}

ultimately show $x \in \mathcal{E} - \mathcal{E}'$ **by** *auto*
qed
moreover have $B \cup (B' - B) = B'$ **using** ‹$B \subseteq B'$› **by** *auto*
ultimately show $B' - B \subseteq \mathcal{E} - \mathcal{E}' \wedge$ *basis-in* $\mathcal{E}$ $(B \cup (B' - B))$
using ‹*basis-in* $\mathcal{E}$ B'› **by** *auto*
qed
qed

lemma *basis-in-supI*:
assumes $B \subseteq \mathcal{E}'$ $\mathcal{E}' \subseteq \mathcal{E}$ $\mathcal{E} \subseteq$ *carrier*
assumes *basis-in* $\mathcal{E}$ B
shows *basis-in* $\mathcal{E}'$ B
proof (*rule basis-inI*)
show $\mathcal{E}' \subseteq$ *carrier* **using** *assms* **by** *auto*
next
show *indep-in* $\mathcal{E}'$ B
proof −
have *indep-in* $\mathcal{E}'$ $(B \cap \mathcal{E}')$
using *assms basis-in-indep-in*[*of* $\mathcal{E}$ B] *indep-in-subI* **by** *auto*
moreover have $B \cap \mathcal{E}' = B$ **using** *assms* **by** *auto*
ultimately show *?thesis* **by** *auto*
qed
next
show $\bigwedge x.\ x \in \mathcal{E}' - B \Longrightarrow \neg$ *indep-in* $\mathcal{E}'$ (*insert* x B)
using *assms basis-in-subset-carrier basis-in-max-indep-in dep-in-subI*[*of* - $\mathcal{E}$ $\mathcal{E}'$]
by *auto*
qed

end

1.5 Relation between dependence and circuits

A set is dependent iff it contains a circuit.

lemma *dep-imp-supset-circuit*:
assumes $X \subseteq$ *carrier*
assumes $\neg$ *indep* X
shows $\exists\, C.$ *circuit* $C \wedge C \subseteq X$
using *assms*
proof (*induction* X *rule*: *remove-induct*)
case (*remove* X)
{
assume $\neg$ *circuit* X
then obtain x **where** $x \in X$ $\neg$ *indep* $(X - \{x\})$
using *remove.prems dep-not-circuit* **by** *auto*
then obtain C **where** *circuit* C $C \subseteq X - \{x\}$
using *remove.prems remove.IH*[*of* x] **by** *auto*
then have *?case* **by** *auto*
}
then show *?case* **using** *remove.prems* **by** *auto*

qed (*auto simp add*: *carrier-finite finite-subset*)

lemma *supset-circuit-imp-dep*:
  assumes *circuit* $C \wedge C \subseteq X$
  shows $\neg$ *indep* X
  using *assms indep-subset circuit-dep* **by** *auto*

lemma *dep-iff-supset-circuit*:
  assumes $X \subseteq$ *carrier*
  shows $\neg$ *indep* $X \longleftrightarrow (\exists C.$ *circuit* $C \wedge C \subseteq X)$
  using *assms dep-imp-supset-circuit supset-circuit-imp-dep* **by** *auto*

context
  fixes $\mathcal{E}$
  assumes $\mathcal{E} \subseteq$ *carrier*
begin

interpretation $\mathcal{E}$: *indep-system* $\mathcal{E}$ *indep-in* $\mathcal{E}$
  using ‹$\mathcal{E} \subseteq$ *carrier*› **by** *auto*

lemma *dep-in-imp-supset-circuit-in*:
  assumes $X \subseteq \mathcal{E}$
  assumes $\neg$ *indep-in* $\mathcal{E}$ X
  shows $\exists C.$ *circuit-in* $\mathcal{E}$ $C \wedge C \subseteq X$
  unfolding *circuit-in-def* **using** $\mathcal{E}$.*dep-imp-supset-circuit*[*OF assms*] **.**

lemma *supset-circuit-in-imp-dep-in*:
  assumes *circuit-in* $\mathcal{E}$ $C \wedge C \subseteq X$
  shows $\neg$ *indep-in* $\mathcal{E}$ X
  using *assms* $\mathcal{E}$.*supset-circuit-imp-dep* **unfolding** *circuit-in-def* **by** *auto*

lemma *dep-in-iff-supset-circuit-in*:
  assumes $X \subseteq \mathcal{E}$
  shows $\neg$ *indep-in* $\mathcal{E}$ $X \longleftrightarrow (\exists C.$ *circuit-in* $\mathcal{E}$ $C \wedge C \subseteq X)$
  using *assms dep-in-imp-supset-circuit-in supset-circuit-in-imp-dep-in* **by** *auto*

end

1.6 Ranks

definition *lower-rank-of* :: *'a set* $\Rightarrow$ *nat* **where**
  lower-rank-of carrier' $\equiv$ *Min* $\{$*card* B $|$ B. *basis-in carrier'* $B\}$

definition *upper-rank-of* :: *'a set* $\Rightarrow$ *nat* **where**
  upper-rank-of carrier' $\equiv$ *Max* $\{$*card* B $|$ B. *basis-in carrier'* $B\}$

lemma *collect-basis-finite*: *finite* (*Collect basis*)
proof −
  have *Collect basis* $\subseteq \{X.\ X \subseteq$ *carrier*$\}$

using *basis-subset-carrier* **by** *auto*
moreover have *finite* ...
using *carrier-finite* **by** *auto*
ultimately show *?thesis* **using** *finite-subset* **by** *auto*
qed

context
fixes $\mathcal{E}$
assumes $*$: $\mathcal{E} \subseteq$ *carrier*
begin

interpretation $\mathcal{E}$: *indep-system* $\mathcal{E}$ *indep-in* $\mathcal{E}$
using $*$ **by** *auto*

lemma *collect-basis-in-finite*: *finite* (*Collect* (*basis-in* $\mathcal{E}$))
unfolding *basis-in-def* **using** $\mathcal{E}$.*collect-basis-finite* **.**

lemma *lower-rank-of-le*: *lower-rank-of* $\mathcal{E} \leq$ *card* $\mathcal{E}$
proof −
have $\exists\, n \in \{card\ B \mid B.\ basis\text{-}in\ \mathcal{E}\ B\}.\ n \leq card\ \mathcal{E}$
using *card-mono*[*OF* $\mathcal{E}$.*carrier-finite* *basis-in-subset-carrier*[*OF* $*$]] *basis-in-ex*[*OF* $*$] **by** *auto*
moreover have *finite* $\{card\ B \mid B.\ basis\text{-}in\ \mathcal{E}\ B\}$
using *collect-basis-in-finite* **by** *auto*
ultimately show *?thesis*
unfolding *lower-rank-of-def* **using** *basis-ex Min-le-iff* **by** *auto*
qed

lemma *upper-rank-of-le*: *upper-rank-of* $\mathcal{E} \leq$ *card* $\mathcal{E}$
proof −
have $\forall\, n \in \{card\ B \mid B.\ basis\text{-}in\ \mathcal{E}\ B\}.\ n \leq card\ \mathcal{E}$
using *card-mono*[*OF* $\mathcal{E}$.*carrier-finite* *basis-in-subset-carrier*[*OF* $*$]] **by** *auto*
then show *?thesis*
unfolding *upper-rank-of-def* **using** *basis-in-ex*[*OF* $*$] *collect-basis-in-finite* **by** *auto*
qed

context
fixes $\mathcal{E}'$
assumes $**$: $\mathcal{E}' \subseteq \mathcal{E}$
begin

interpretation $\mathcal{E}'_1$: *indep-system* $\mathcal{E}'$ *indep-in* $\mathcal{E}'$
using $*$ $**$ **by** *auto*
interpretation $\mathcal{E}'_2$: *indep-system* $\mathcal{E}'$ $\mathcal{E}$.*indep-in* $\mathcal{E}'$
using $*$ $**$ **by** *auto*

lemma *lower-rank-of-sub-cong*:
shows $\mathcal{E}$.*lower-rank-of* $\mathcal{E}'$ = *lower-rank-of* $\mathcal{E}'$

proof −
  have $\bigwedge B.\ \mathcal{E}'_1.basis\ B \longleftrightarrow \mathcal{E}'_2.basis\ B$
    using $**$ *basis-in-sub-cong*$[OF\ *,\ of\ \mathcal{E}']$
    unfolding *basis-in-def* $\mathcal{E}$*.basis-in-def* **by** *auto*
  then show *?thesis*
    unfolding *lower-rank-of-def* $\mathcal{E}$*.lower-rank-of-def*
    using *basis-in-sub-cong*$[OF\ *\ **]$
    by *auto*
qed

lemma *upper-rank-of-sub-cong*:
  shows $\mathcal{E}$*.upper-rank-of* $\mathcal{E}'$ = *upper-rank-of* $\mathcal{E}'$
proof −
  have $\bigwedge B.\ \mathcal{E}'_1.basis\ B \longleftrightarrow \mathcal{E}'_2.basis\ B$
    using $**$ *basis-in-sub-cong*$[OF\ *,\ of\ \mathcal{E}']$
    unfolding *basis-in-def* $\mathcal{E}$*.basis-in-def* **by** *auto*
  then show *?thesis*
    unfolding *upper-rank-of-def* $\mathcal{E}$*.upper-rank-of-def*
    using *basis-in-sub-cong*$[OF\ *\ **]$
    by *auto*
qed

end

end

end

end

2 Matroids

theory *Matroid*
  imports *Indep-System*
begin

lemma *card-subset-ex*:
  assumes *finite A* $n \leq$ *card A*
  shows $\exists B \subseteq A.$ *card B* $= n$
using *assms*
proof (*induction A arbitrary*: *n rule*: *finite-induct*)
  case (*insert x A*)
  show *?case*
  proof (*cases n*)
    case *0*
    then show *?thesis* **using** *card.empty* **by** *blast*
  next
    case (*Suc k*)
    then have $\exists B \subseteq A.$ *card B* $= k$ **using** *insert* **by** *auto*

then obtain B **where** $B \subseteq A$ *card* $B = k$ **by** *auto*
moreover from *this* **have** *finite* B **using** *insert.hyps finite-subset* **by** *auto*
ultimately have *card* (*insert* x B) $= n$
using *Suc insert.hyps card-insert-disjoint* **by** *fastforce*
then show *?thesis* **using** ‹$B \subseteq A$› **by** *blast*
qed
qed *auto*

locale *matroid* = *indep-system* +
assumes *augment-aux*:
indep $X \Longrightarrow$ *indep* $Y \Longrightarrow$ *card* X = *Suc* (*card* Y) $\Longrightarrow \exists x \in X - Y$. *indep* (*insert* x Y)
begin

lemma *augment*:
assumes *indep* X *indep* Y *card* $Y <$ *card* X
shows $\exists x \in X - Y$. *indep* (*insert* x Y)
proof −
obtain X' **where** $X' \subseteq X$ *card* X' = *Suc* (*card* Y)
using *assms card-subset-ex*[*of* X *Suc* (*card* Y)] *indep-finite* **by** *auto*
then obtain x **where** $x \in X' - Y$ *indep* (*insert* x Y)
using *assms augment-aux*[*of* X' Y] *indep-subset* **by** *auto*
then show *?thesis* **using** ‹$X' \subseteq X$› **by** *auto*
qed

lemma *augment-psubset*:
assumes *indep* X *indep* Y $Y \subset X$
shows $\exists x \in X - Y$. *indep* (*insert* x Y)
using *assms augment psubset-card-mono indep-finite* **by** *blast*

2.1 Minors

A subset of the ground set induces a matroid.

lemma *matroid-subset* [*simp*, *intro*]:
assumes $\mathcal{E} \subseteq$ *carrier*
shows *matroid* $\mathcal{E}$ (*indep-in* $\mathcal{E}$)
unfolding *matroid-def matroid-axioms-def*
proof (*safe*, *goal-cases indep-system augment*)
case *indep-system*
then show *?case* **using** *indep-system-subset*[*OF assms*] **.**
next
case (*augment* X Y)
then show *?case* **using** *augment-aux*[*of* X Y] **unfolding** *indep-in-def* **by** *auto*
qed

context
fixes $\mathcal{E}$
assumes $\mathcal{E} \subseteq$ *carrier*
begin

interpretation $\mathcal{E}$: *matroid* $\mathcal{E}$ *indep-in* $\mathcal{E}$
  using ‹$\mathcal{E} \subseteq$ *carrier*› **by** *auto*

lemmas *augment-aux-indep-in* = $\mathcal{E}$.*augment-aux*
lemmas *augment-indep-in* = $\mathcal{E}$.*augment*
lemmas *augment-psubset-indep-in* = $\mathcal{E}$.*augment-psubset*

end

2.2 Bases

lemma *basis-card*:
  assumes *basis* B_1
  assumes *basis* B_2
  shows *card* B_1 = *card* B_2
proof (*rule ccontr*, *goal-cases False*)
  case *False*
  then have *card* B_1 < *card* $B_2 \vee$ *card* B_2 < *card* B_1 **by** *auto*
  moreover {
    fix B_1 B_2
    assume *basis* B_1 *basis* B_2 *card* B_1 < *card* B_2
    then obtain x **where** $x \in B_2 - B_1$ *indep* (*insert* x B_1)
      using *augment basisD* **by** *blast*
    then have $x \in$ *carrier* $- B_1$
      using ‹*basis* B_1› *basisD indep-subset-carrier* **by** *blast*
    then have ¬ *indep* (*insert* x B_1) **using** ‹*basis* B_1› *basisD* **by** *auto*
    then have *False* **using** ‹*indep* (*insert* x B_1)› **by** *auto*
  }
  ultimately show *?case* **using** *assms* **by** *auto*
qed

lemma *basis-indep-card*:
  assumes *indep* X
  assumes *basis* B
  shows *card* $X \leq$ *card* B
proof −
  obtain B' **where** *basis* B' $X \subseteq B'$ **using** *assms indep-imp-subset-basis* **by** *auto*
  then show *?thesis* **using** *assms basis-finite basis-card*[*of* B B'] **by** (*auto intro*: *card-mono*)
qed

lemma *basis-augment*:
  assumes *basis* B_1 *basis* B_2 $x \in B_1 - B_2$
  shows $\exists\, y \in B_2 - B_1$. *basis* (*insert* y ($B_1 - \{x\}$))
proof −
  let *?*B_1 = $B_1 - \{x\}$
  have *card ?*B_1 < *card* B_2
    using *assms basis-card*[*of* B_1 B_2] *card-Diff1-less*[*OF basis-finite*, *of* B_1] **by** *auto*

moreover have *indep ?B*$_1$ **using** *assms basis-indep*[*of B*$_1$] *indep-subset*[*of B*$_1$ *?B*$_1$] **by** *auto*
ultimately obtain *y* **where** *y*: $y \in B_2 - ?B_1$ *indep* (*insert y ?B*$_1$)
using *assms augment*[*of B*$_2$ *?B*$_1$] *basis-indep* **by** *auto*
let *?B*$_1$′ = *insert y ?B*$_1$
have *basis ?B*$_1$′ **using** ‹*indep ?B*$_1$′›
proof (*rule basisI*, *goal-cases insert*)
case (*insert x*)
have *card* (*insert x ?B*$_1$′) > *card B*$_1$
proof −
have *card* (*insert x ?B*$_1$′) = *Suc* (*card ?B*$_1$′)
using *insert card.insert-remove*[*OF indep-finite*, *of ?B*$_1$′] *y* **by** *auto*
also have ... = *Suc* (*Suc* (*card ?B*$_1$))
using *card.insert-remove*[*OF indep-finite*, *of ?B*$_1$] ‹*indep ?B*$_1$› *y* **by** *auto*
also have ... = *Suc* (*card B*$_1$)
using *assms basis-finite*[*of B*$_1$] *card.remove*[*of B*$_1$] **by** *auto*
finally show *?thesis* **by** *auto*
qed
then have ¬*indep* (*insert x* (*insert y ?B*$_1$))
using *assms basis-indep-card*[*of insert x* (*insert y ?B*$_1$) *B*$_1$] **by** *auto*
moreover have *insert x* (*insert y ?B*$_1$) ⊆ *carrier*
using *assms insert y basis-finite indep-subset-carrier* **by** *auto*
ultimately show *?case* **by** *auto*
qed
then show *?thesis* **using** *assms y* **by** *auto*
qed

context
fixes $\mathcal{E}$
assumes ∗: $\mathcal{E} \subseteq$ *carrier*
begin

interpretation $\mathcal{E}$: *matroid* $\mathcal{E}$ *indep-in* $\mathcal{E}$
using ‹$\mathcal{E} \subseteq$ *carrier*› **by** *auto*

lemmas *basis-in-card* = $\mathcal{E}$.*basis-card*[*OF basis-inD-aux*[*OF* ∗] *basis-inD-aux*[*OF* ∗]]
lemmas *basis-in-indep-in-card* = $\mathcal{E}$.*basis-indep-card*[*OF* - *basis-inD-aux*[*OF* ∗]]

lemma *basis-in-augment*:
assumes *basis-in* $\mathcal{E}$ *B*$_1$ *basis-in* $\mathcal{E}$ *B*$_2$ $x \in B_1 - B_2$
shows $\exists y \in B_2 - B_1$. *basis-in* $\mathcal{E}$ (*insert y* ($B_1 - \{x\}$))
using *assms* $\mathcal{E}$.*basis-augment* **unfolding** *basis-in-def* **by** *auto*

end

2.3 Circuits

lemma *circuit-elim*:

assumes *circuit* C_1 *circuit* C_2 $C_1 \neq C_2$ $x \in C_1 \cap C_2$
shows $\exists\, C_3 \subseteq (C_1 \cup C_2) - \{x\}$. *circuit* C_3
proof −
let *?C* $= (C_1 \cup C_2) - \{x\}$
let *?carrier* $= C_1 \cup C_2$

have *assms'*: *circuit-in carrier* C_1 *circuit-in carrier* C_2
using *assms circuit-imp-circuit-in* **by** *auto*

have *?C* $\subseteq$ *carrier* **using** *assms circuit-subset-carrier* **by** *auto*
show *?thesis*
proof (*cases indep ?C*)
case *False*
then show *?thesis* **using** *dep-iff-supset-circuit* ‹*?C* $\subseteq$ *carrier*› **by** *auto*
next
case *True*
then have *indep-in ?carrier ?C* **using** ‹*?C* $\subseteq$ *carrier*› **by** (*auto intro*: *indep-inI*)

have $*$: *?carrier* $\subseteq$ *carrier* **using** *assms circuit-subset-carrier* **by** *auto*
obtain *y* **where** *y*: $y \in C_2$ $y \notin C_1$ **using** *assms circuit-subset-eq* **by** *blast*
then have *indep-in ?carrier* $(C_2 - \{y\})$
using *assms'* *circuit-in-min-dep-in*[*OF* $*$ *circuit-in-supI*[*OF* $*$, *of* C_2]] **by** *auto*
then obtain *B* **where** *B*: *basis-in ?carrier B* $C_2 - \{y\} \subseteq B$
using $*$ *assms indep-in-imp-subset-basis-in*[*of ?carrier* $C_2 - \{y\}$] **by** *auto*

have $y \notin B$
proof (*rule ccontr*, *goal-cases False*)
case *False*
then have $C_2 \subseteq B$ **using** *B* **by** *auto*
moreover have *circuit-in ?carrier* C_2 **using** $*$ *assms'* *circuit-in-supI* **by** *auto*
ultimately have ¬ *indep-in ?carrier B*
using *B basis-in-subset-carrier*[*OF* $*$] *supset-circuit-in-imp-dep-in*[*OF* $*$] **by** *auto*
then show *False* **using** *assms B basis-in-indep-in*[*OF* $*$] **by** *auto*
qed

have $C_1 - B \neq \{\}$
proof (*rule ccontr*, *goal-cases False*)
case *False*
then have $C_1 - (C_1 \cap B) = \{\}$ **by** *auto*
then have $C_1 = C_1 \cap B$ **using** *assms circuit-subset-eq* **by** *auto*
moreover have *indep* $(C_1 \cap B)$
using *assms B basis-in-indep-in*[*OF* $*$] *indep-in-subset*[*OF* $*$, *of B* $C_1 \cap B$] *indep-in-indep*
by *auto*
ultimately show *?case* **using** *assms circuitD* **by** *auto*
qed
then obtain *z* **where** *z*: $z \in C_1$ $z \notin B$ **by** *auto*

have $y \neq z$ **using** y z **by** *auto*
have $x \in C_1$ $x \in C_2$ **using** *assms* **by** *auto*

have *finite ?carrier* **using** *assms carrier-finite finite-subset* **by** *auto*
have $\mathit{card}\ B \leq \mathit{card}\ (\mathit{?carrier} - \{y, z\})$
proof (*rule card-mono*)
show $\mathit{finite}\ (C_1 \cup C_2 - \{y, z\})$ **using** ‹*finite ?carrier*› **by** *auto*
next
show $B \subseteq C_1 \cup C_2 - \{y, z\}$
using *B basis-in-subset-carrier*[*OF* $*$, *of B*] ‹$y \notin B$› ‹$z \notin B$› **by** *auto*
qed
also have $\ldots = \mathit{card}\ \mathit{?carrier} - \mathit{2}$
using ‹*finite ?carrier*› ‹$y \in C_2$› ‹$z \in C_1$› ‹$y \neq z$› *card-Diff-subset-Int* **by** *auto*
also have $\ldots < \mathit{card}\ \mathit{?carrier} - \mathit{1}$
proof −
have $\mathit{card}\ \mathit{?carrier} = \mathit{card}\ C_1 + \mathit{card}\ C_2 - \mathit{card}\ (C_1 \cap C_2)$
using *assms* ‹*finite ?carrier*› *card-Un-Int*[*of* C_1 C_2] **by** *auto*
also have $\ldots = \mathit{card}\ C_1 + (\mathit{card}\ C_2 - \mathit{card}\ (C_1 \cap C_2))$
using *assms* ‹*finite ?carrier*› *card-mono*[*of* C_2] **by** *auto*
also have $\ldots = \mathit{card}\ C_1 + \mathit{card}\ (C_2 - C_1)$
proof −
have $\mathit{card}\ (C_2 - C_1) = \mathit{card}\ C_2 - \mathit{card}\ (C_2 \cap C_1)$
using *assms* ‹*finite ?carrier*› *card-Diff-subset-Int*[*of* C_2 C_1] **by** *auto*
also have $\ldots = \mathit{card}\ C_2 - \mathit{card}\ (C_1 \cap C_2)$ **by** (*simp add*: *inf-commute*)
finally show *?thesis* **by** *auto*
qed
finally have $\mathit{card}\ (C_1 \cup C_2) = \mathit{card}\ C_1 + \mathit{card}\ (C_2 - C_1)$ **.**
moreover have $\mathit{card}\ C_1 > \mathit{0}$ **using** *assms circuit-nonempty* ‹*finite ?carrier*› **by** *auto*
moreover have $\mathit{card}\ (C_2 - C_1) > \mathit{0}$ **using** *assms* ‹*finite ?carrier*› ‹$y \in C_2$› ‹$y \notin C_1$› **by** *auto*
ultimately show *?thesis* **by** *auto*
qed
also have $\ldots = \mathit{card}\ \mathit{?C}$
using ‹*finite ?carrier*› *card-Diff-singleton* ‹$x \in C_1$› ‹$x \in C_2$› **by** *auto*
finally have $\mathit{card}\ B < \mathit{card}\ \mathit{?C}$ **.**
then have *False*
using *basis-in-indep-in-card*[*OF* $*$, *of ?C B*] *B* ‹*indep-in ?carrier ?C*› **by** *auto*
then show *?thesis* **by** *auto*
qed
qed

lemma *min-dep-imp-supset-circuit*:
assumes *indep X*
assumes *circuit C*
assumes $C \subseteq \mathit{insert}\ x\ X$
shows $x \in C$

proof (*rule ccontr*)
  assume $x \notin C$
  then have $C \subseteq X$ **using** *assms* **by** *auto*
  then have *indep* C **using** *assms indep-subset* **by** *auto*
  then show *False* **using** *assms circuitD* **by** *auto*
qed

lemma *min-dep-imp-ex1-supset-circuit*:
  assumes $x \in$ *carrier*
  assumes *indep* X
  assumes $\neg$ *indep* (*insert* x X)
  shows $\exists!C.$ *circuit* $C \wedge C \subseteq$ *insert* x X
proof $-$
  obtain C **where** C: *circuit* C $C \subseteq$ *insert* x X
    using *assms indep-subset-carrier dep-iff-supset-circuit* **by** *auto*

  show *?thesis*
  proof (*rule ex1I*, *goal-cases ex unique*)
    show *circuit* $C \wedge C \subseteq$ *insert* x X **using** C **by** *auto*
  next
    {
      fix C'
      assume C': *circuit* C' $C' \subseteq$ *insert* x X
      have $C' = C$
      proof (*rule ccontr*)
        assume $C' \neq C$
        moreover have $x \in C' \cap C$ **using** C C' *assms min-dep-imp-supset-circuit* **by** *auto*
        ultimately have $\neg$ *indep* $(C' \cup C - \{x\})$
          using *circuit-elim*[*OF* $C(1)$ $C'(1)$, *of* x] *supset-circuit-imp-dep*[*of* - $C' \cup C - \{x\}$] **by** *auto*
        moreover have $C' \cup C - \{x\} \subseteq X$ **using** C C' **by** *auto*
        ultimately show *False* **using** *assms indep-subset* **by** *auto*
      qed
    }
    then show $\bigwedge C'.$ *circuit* $C' \wedge C' \subseteq$ *insert* x $X \Longrightarrow C' = C$
      by *auto*
  qed
qed

lemma *basis-ex1-supset-circuit*:
  assumes *basis* B
  assumes $x \in$ *carrier* $- B$
  shows $\exists!C.$ *circuit* $C \wedge C \subseteq$ *insert* x B
  using *assms min-dep-imp-ex1-supset-circuit basisD* **by** *auto*

definition *fund-circuit* :: $'a \Rightarrow 'a$ *set* $\Rightarrow 'a$ *set* **where**
  fund-circuit x $B \equiv$ (*THE* C. *circuit* $C \wedge C \subseteq$ *insert* x B)

```
lemma circuit-iff-fund-circuit:
  circuit C ⟷ (∃x B. x ∈ carrier − B ∧ basis B ∧ C = fund-circuit x B)
proof (safe, goal-cases LTR RTL)
  case LTR
  then obtain x where x ∈ C using circuit-nonempty by auto
  then have indep (C − {x}) using LTR unfolding circuit-def by auto
  then obtain B where B: basis B C − {x} ⊆ B using indep-imp-subset-basis
by auto
  then have x ∈ carrier using LTR circuit-subset-carrier ‹x ∈ C› by auto
  moreover have x ∉ B
  proof (rule ccontr, goal-cases False)
    case False
    then have C ⊆ B using ‹C − {x} ⊆ B› by auto
    then have ¬ indep B using LTR B basis-subset-carrier supset-circuit-imp-dep
by auto
    then show ?case using B basis-indep by auto
  qed
  ultimately show ?case
    unfolding fund-circuit-def
    using LTR B theI-unique[OF basis-ex1-supset-circuit[of B x], of C] by auto
next
  case (RTL x B)
  then have ∃!C. circuit C ∧ C ⊆ insert x B
    using min-dep-imp-ex1-supset-circuit basisD[of B] by auto
  then show ?case
    unfolding fund-circuit-def
    using theI[of λC. circuit C ∧ C ⊆ insert x B] by fastforce
qed

lemma fund-circuitI:
  assumes basis B
  assumes x ∈ carrier − B
  assumes circuit C
  assumes C ⊆ insert x B
  shows fund-circuit x B = C
  unfolding fund-circuit-def
  using assms theI-unique[OF basis-ex1-supset-circuit, of B x C] by auto

definition fund-circuit-in where fund-circuit-in ℰ x B ≡ matroid.fund-circuit ℰ
(indep-in ℰ) x B

context
  fixes ℰ
  assumes *: ℰ ⊆ carrier
begin

interpretation ℰ: matroid ℰ indep-in ℰ
  using ‹ℰ ⊆ carrier› by auto
```

lemma *fund-circuit-inI-aux*: $\mathcal{E}$*.fund-circuit* x B = *fund-circuit-in* $\mathcal{E}$ x B
unfolding *fund-circuit-in-def* **by** *auto*

lemma *circuit-in-elim*:
assumes *circuit-in* $\mathcal{E}$ C_1 *circuit-in* $\mathcal{E}$ C_2 $C_1 \neq C_2$ $x \in C_1 \cap C_2$
shows $\exists\, C_3 \subseteq (C_1 \cup C_2) - \{x\}$. *circuit-in* $\mathcal{E}$ C_3
using *assms* $\mathcal{E}$*.circuit-elim* **unfolding** *circuit-in-def* **by** *auto*

lemmas *min-dep-in-imp-supset-circuit-in* = $\mathcal{E}$*.min-dep-imp-supset-circuit*[*OF* - *circuit-inD-aux*[*OF* ∗]]

lemma *min-dep-in-imp-ex1-supset-circuit-in*:
assumes $x \in \mathcal{E}$
assumes *indep-in* $\mathcal{E}$ X
assumes ¬ *indep-in* $\mathcal{E}$ (*insert* x X)
shows $\exists!C$. *circuit-in* $\mathcal{E}$ C ∧ $C \subseteq$ *insert* x X
using *assms* $\mathcal{E}$*.min-dep-imp-ex1-supset-circuit* **unfolding** *circuit-in-def* **by** *auto*

lemma *basis-in-ex1-supset-circuit-in*:
assumes *basis-in* $\mathcal{E}$ B
assumes $x \in \mathcal{E} - B$
shows $\exists!C$. *circuit-in* $\mathcal{E}$ C ∧ $C \subseteq$ *insert* x B
using *assms* $\mathcal{E}$*.basis-ex1-supset-circuit* **unfolding** *circuit-in-def basis-in-def* **by** *auto*

lemma *fund-circuit-inI*:
assumes *basis-in* $\mathcal{E}$ B
assumes $x \in \mathcal{E} - B$
assumes *circuit-in* $\mathcal{E}$ C
assumes $C \subseteq$ *insert* x B
shows *fund-circuit-in* $\mathcal{E}$ x B = C
using *assms* $\mathcal{E}$*.fund-circuitI*
unfolding *basis-in-def circuit-in-def fund-circuit-in-def* **by** *auto*

end

context
fixes $\mathcal{E}$
assumes ∗: $\mathcal{E} \subseteq$ *carrier*
begin

interpretation $\mathcal{E}$: *matroid* $\mathcal{E}$ *indep-in* $\mathcal{E}$
using ‹$\mathcal{E} \subseteq$ *carrier*› **by** *auto*

lemma *fund-circuit-in-sub-cong*:
assumes $\mathcal{E}' \subseteq \mathcal{E}$
assumes $x \in \mathcal{E}' - B$
assumes *basis-in* $\mathcal{E}'$ B
shows $\mathcal{E}$*.fund-circuit-in* $\mathcal{E}'$ x B = *fund-circuit-in* $\mathcal{E}'$ x B

proof −
  obtain *C* **where** *C*: *circuit-in* $\mathcal{E}'$ *C* $C \subseteq$ *insert x B*
    using ∗ *assms basis-in-ex1-supset-circuit-in*[*of* $\mathcal{E}'$ *B x*] **by** *auto*
  then have *fund-circuit-in* $\mathcal{E}'$ *x B* = *C*
    using ∗ *assms fund-circuit-inI* **by** *auto*
  also have ... = $\mathcal{E}$.*fund-circuit-in* $\mathcal{E}'$ *x B*
   using ∗ *assms C* $\mathcal{E}$.*fund-circuit-inI basis-in-sub-cong*[*of* $\mathcal{E}$] *circuit-in-sub-cong*[*of* $\mathcal{E}$] **by** *auto*
  finally show *?thesis* **by** *auto*
qed

end

2.4 Ranks

abbreviation *rank-of* **where** *rank-of* ≡ *lower-rank-of*

lemmas *rank-of-def* = *lower-rank-of-def*
lemmas *rank-of-sub-cong* = *lower-rank-of-sub-cong*
lemmas *rank-of-le* = *lower-rank-of-le*

context
  fixes $\mathcal{E}$
  assumes ∗: $\mathcal{E} \subseteq$ *carrier*
begin

interpretation $\mathcal{E}$: *matroid* $\mathcal{E}$ *indep-in* $\mathcal{E}$
  using ∗ **by** *auto*

lemma *lower-rank-of-eq-upper-rank-of*: *lower-rank-of* $\mathcal{E}$ = *upper-rank-of* $\mathcal{E}$
proof −
  obtain *B* **where** *basis-in* $\mathcal{E}$ *B* **using** *basis-in-ex*[*OF* ∗] **by** *auto*
  then have {*card B* | *B*. *basis-in* $\mathcal{E}$ *B*} = {*card B*}
    by *safe* (*auto dest*: *basis-in-card*[*OF* ∗])
  then show *?thesis* **unfolding** *lower-rank-of-def upper-rank-of-def* **by** *auto*
qed

lemma *rank-of-eq-card-basis-in*:
  assumes *basis-in* $\mathcal{E}$ *B*
  shows *rank-of* $\mathcal{E}$ = *card B*
proof −
  have {*card B* | *B*. *basis-in* $\mathcal{E}$ *B*} = {*card B*} **using** *assms* **by** *safe* (*auto dest*: *basis-in-card*[*OF* ∗])
  then show *?thesis* **unfolding** *rank-of-def* **by** *auto*
qed

lemma *rank-of-indep-in-le*:
  assumes *indep-in* $\mathcal{E}$ *X*
  shows *card X* ≤ *rank-of* $\mathcal{E}$

proof −
  {
    fix B
    assume *basis-in* $\mathcal{E}$ B
    moreover obtain B' **where** *basis-in* $\mathcal{E}$ B' $X \subseteq B'$
      using *assms indep-in-imp-subset-basis-in*[*OF* ∗] **by** *auto*
    ultimately have *card* $X \leq$ *card* B
      using *card-mono*[*OF basis-in-finite*[*OF* ∗]] *basis-in-card*[*OF* ∗, *of* B B'] **by** *auto*
  }
  moreover have *finite* {*card* B | B. *basis-in* $\mathcal{E}$ B}
    using *collect-basis-in-finite*[*OF* ∗] **by** *auto*
  ultimately show *?thesis*
    unfolding *rank-of-def* **using** *basis-in-ex*[*OF* ∗] **by** *auto*
qed

end

lemma *rank-of-mono*:
  assumes $X \subseteq Y$
  assumes $Y \subseteq$ *carrier*
  shows *rank-of* $X \leq$ *rank-of* Y
proof −
  obtain B_X **where** B_X: *basis-in* X B_X **using** *assms basis-in-ex*[*of* X] **by** *auto*
  moreover obtain B_Y **where** B_Y: *basis-in* Y B_Y **using** *assms basis-in-ex*[*of* Y] **by** *auto*
  moreover have *card* $B_X \leq$ *card* B_Y
    using *assms basis-in-indep-in-card*[*OF* - - B_Y] *basis-in-indep-in*[*OF* - B_X] *indep-in-subI-subset*
    by *auto*
  ultimately show *?thesis* **using** *assms rank-of-eq-card-basis-in* **by** *auto*
qed

lemma *rank-of-insert-le*:
  assumes $X \subseteq$ *carrier*
  assumes $x \in$ *carrier*
  shows *rank-of* (*insert* x X) $\leq$ *Suc* (*rank-of* X)
proof −
  obtain B **where** B: *basis-in* X B **using** *assms basis-in-ex*[*of* X] **by** *auto*
  have *basis-in* (*insert* x X) B ∨ *basis-in* (*insert* x X) (*insert* x B)
  proof −
    obtain B' **where** B': $B' \subseteq$ *insert* x $X - X$ *basis-in* (*insert* x X) ($B \cup B'$)
      using *assms* B *basis-in-subI*[*of insert* x X X B] **by** *auto*
    then have $B' = \{\} \vee B' = \{x\}$ **by** *auto*
    then show *?thesis*
    proof
      assume $B' = \{\}$
      then have *basis-in* (*insert* x X) B **using** B' **by** *auto*
      then show *?thesis* **by** *auto*

next
  assume $B' = \{x\}$
  then have *basis-in* (*insert x X*) (*insert x B*) **using** B' **by** *auto*
  then show *?thesis* **by** *auto*
qed
qed
then show *?thesis*
proof
  assume *basis-in* (*insert x X*) *B*
  then show *?thesis*
    using *assms B rank-of-eq-card-basis-in* **by** *auto*
next
  assume *basis-in* (*insert x X*) (*insert x B*)
  then have *rank-of* (*insert x X*) = *card* (*insert x B*)
    using *assms rank-of-eq-card-basis-in* **by** *auto*
  also have ... = *Suc* (*card* ($B - \{x\}$))
    using *assms card.insert-remove*[*of B x*] **using** *B basis-in-finite* **by** *auto*
  also have ... $\leq$ *Suc* (*card B*)
    using *assms B basis-in-finite card-Diff1-le*[*of B*] **by** *auto*
  also have ... = *Suc* (*rank-of X*)
    using *assms B rank-of-eq-card-basis-in* **by** *auto*
  finally show *?thesis* **.**
qed
qed

lemma *rank-of-Un-Int-le*:
  assumes $X \subseteq$ *carrier*
  assumes $Y \subseteq$ *carrier*
  shows *rank-of* ($X \cup Y$) + *rank-of* ($X \cap Y$) $\leq$ *rank-of X* + *rank-of Y*
proof –
  obtain *B-Int* **where** *B-Int*: *basis-in* ($X \cap Y$) *B-Int* **using** *assms basis-in-ex*[*of* $X \cap Y$] **by** *auto*
  then have *indep-in* ($X \cup Y$) *B-Int*
    using *assms indep-in-subI-subset*[*OF - basis-in-indep-in*[*of* $X \cap Y$ *B-Int*], *of* $X \cup Y$] **by** *auto*
  then obtain *B-Un* **where** *B-Un*: *basis-in* ($X \cup Y$) *B-Un B-Int* $\subseteq$ *B-Un*
    using *assms indep-in-imp-subset-basis-in*[*of* $X \cup Y$ *B-Int*] **by** *auto*

  have *card* (*B-Un* $\cap$ ($X \cup Y$)) + *card* (*B-Un* $\cap$ ($X \cap Y$)) = *card* ((*B-Un* $\cap$ *X*) $\cup$ (*B-Un* $\cap$ *Y*)) + *card* ((*B-Un* $\cap$ *X*) $\cap$ (*B-Un* $\cap$ *Y*))
    by (*simp add*: *inf-assoc inf-left-commute inf-sup-distrib1*)
  also have ... = *card* (*B-Un* $\cap$ *X*) + *card* (*B-Un* $\cap$ *Y*)
  proof –
    have *finite* (*B-Un* $\cap$ *X*) *finite* (*B-Un* $\cap$ *Y*)
      using *assms finite-subset*[*OF - carrier-finite*] **by** *auto*
    then show *?thesis* **using** *card-Un-Int*[*of B-Un* $\cap$ *X B-Un* $\cap$ *Y*] **by** *auto*
  qed
  also have ... $\leq$ *rank-of X* + *rank-of Y*
  proof –

have *card* $(B\text{-}Un \cap X) \leq$ *rank-of X*
proof −
have *indep-in X* $(B\text{-}Un \cap X)$ **using** *assms basis-in-indep-in*[*OF* - *B-Un*(*1*)] *indep-in-subI* **by** *auto*
then show *?thesis* **using** *assms rank-of-indep-in-le* **by** *auto*
qed
moreover have *card* $(B\text{-}Un \cap Y) \leq$ *rank-of Y*
proof −
have *indep-in Y* $(B\text{-}Un \cap Y)$ **using** *assms basis-in-indep-in*[*OF* - *B-Un*(*1*)] *indep-in-subI* **by** *auto*
then show *?thesis* **using** *assms rank-of-indep-in-le* **by** *auto*
qed
ultimately show *?thesis* **by** *auto*
qed
finally have *rank-of X* + *rank-of Y* $\geq$ *card* $(B\text{-}Un \cap (X \cup Y))$ + *card* $(B\text{-}Un \cap (X \cap Y))$ **.**
moreover have $B\text{-}Un \cap (X \cup Y) = B\text{-}Un$ **using** *assms basis-in-subset-carrier*[*OF* - *B-Un*(*1*)] **by** *auto*
moreover have $B\text{-}Un \cap (X \cap Y) = B\text{-}Int$
proof −
have *card* $(B\text{-}Un \cap (X \cap Y)) \leq$ *card B-Int*
proof −
have *indep-in* $(X \cap Y)$ $(B\text{-}Un \cap (X \cap Y))$
using *assms basis-in-indep-in*[*OF* - *B-Un*(*1*)] *indep-in-subI* **by** *auto*
then show *?thesis* **using** *assms basis-in-indep-in-card*[*of* $X \cap Y$ - *B-Int*] *B-Int* **by** *auto*
qed
moreover have *finite* $(B\text{-}Un \cap (X \cap Y))$
using *assms carrier-finite finite-subset*[*of* $B\text{-}Un \cap (X \cap Y)$] **by** *auto*
moreover have $B\text{-}Int \subseteq B\text{-}Un \cap (X \cap Y)$
using *assms B-Un B-Int basis-in-subset-carrier*[*of* $X \cap Y$ *B-Int*] **by** *auto*
ultimately show *?thesis* **using** *card-seteq* **by** *blast*
qed
ultimately have *rank-of X* + *rank-of Y* $\geq$ *card B-Un* + *card B-Int* **by** *auto*
moreover have *card B-Un* = *rank-of* $(X \cup Y)$
using *assms rank-of-eq-card-basis-in*[*OF* - *B-Un*(*1*)] **by** *auto*
moreover have *card B-Int* = *rank-of* $(X \cap Y)$
using *assms rank-of-eq-card-basis-in*[*OF* - *B-Int*] **by** *fastforce*
ultimately show *rank-of X* + *rank-of Y* $\geq$ *rank-of* $(X \cup Y)$ + *rank-of* $(X \cap Y)$ **by** *auto*
qed

lemma *rank-of-Un-absorbI*:
assumes $X \subseteq$ *carrier* $Y \subseteq$ *carrier*
assumes $\bigwedge y.\ y \in Y - X \Longrightarrow$ *rank-of* (*insert y X*) = *rank-of X*
shows *rank-of* $(X \cup Y)$ = *rank-of X*
proof −
have *finite* $(Y - X)$ **using** *finite-subset*[*OF* ‹$Y \subseteq$ *carrier*›] *carrier-finite* **by** *auto*

then show *?thesis* **using** *assms*
proof (*induction* $Y - X$ *arbitrary*: Y *rule*: *finite-induct*)
case *empty*
then have $X \cup Y = X$ **by** *auto*
then show *?case* **by** *auto*
next
case (*insert* y F)
have *rank-of* $(X \cup Y)$ + *rank-of* $X \leq$ *rank-of* X + *rank-of* X
proof −
have *rank-of* $(X \cup Y)$ + *rank-of* X = *rank-of* $((X \cup (Y - \{y\})) \cup$ (*insert* y X)) + *rank-of* $((X \cup (Y - \{y\})) \cap$ (*insert* y X))
proof −
have $X \cup Y = (X \cup (Y - \{y\})) \cup$ (*insert* y X) $X = (X \cup (Y - \{y\})) \cap$ (*insert* y X) **using** *insert* **by** *auto*
then show *?thesis* **by** *auto*
qed
also have ... $\leq$ *rank-of* $(X \cup (Y - \{y\}))$ + *rank-of* (*insert* y X)
proof (*rule rank-of-Un-Int-le*)
show $X \cup (Y - \{y\}) \subseteq$ *carrier* **using** *insert* **by** *auto*
next
show *insert* y $X \subseteq$ *carrier* **using** *insert* **by** *auto*
qed
also have ... = *rank-of* $(X \cup (Y - \{y\}))$ + *rank-of* X
proof −
have $y \in Y - X$ **using** *insert* **by** *auto*
then show *?thesis* **using** *insert* **by** *auto*
qed
also have ... = *rank-of* X + *rank-of* X
proof −
have $F = (Y - \{y\}) - X$ $Y - \{y\} \subseteq$ *carrier* **using** *insert* **by** *auto*
then show *?thesis* **using** *insert insert*(*3*)[*of* $Y - \{y\}$] **by** *auto*
qed
finally show *?thesis* **.**
qed
moreover have *rank-of* $(X \cup Y)$ + *rank-of* $X \geq$ *rank-of* X + *rank-of* X
using *insert rank-of-mono* **by** *auto*
ultimately show *?case* **by** *auto*
qed
qed

lemma *indep-iff-rank-of*:
assumes $X \subseteq$ *carrier*
shows *indep* $X \longleftrightarrow$ *rank-of* X = *card* X
proof (*standard*, *goal-cases LTR RTL*)
case *LTR*
then have *indep-in* X X **by** (*auto intro*: *indep-inI*)
then have *basis-in* X X **by** (*auto intro*: *basis-inI*[*OF assms*])
then show *?case* **using** *rank-of-eq-card-basis-in*[*OF assms*] **by** *auto*
next

case *RTL*
obtain *B* **where** *B*: *basis-in X B* **using** *basis-in-ex*[*OF assms*] **by** *auto*
then have *card B* = *card X* **using** *RTL rank-of-eq-card-basis-in*[*OF assms*] **by** *auto*
then have *B* = *X*
using *basis-in-subset-carrier*[*OF assms B*] *card-seteq*[*OF finite-subset*[*OF assms carrier-finite*]]
by *auto*
then show *?case* **using** *basis-in-indep-in*[*OF assms B*] *indep-in-indep* **by** *auto*
qed

lemma *basis-iff-rank-of*:
assumes $X \subseteq$ *carrier*
shows *basis X* $\longleftrightarrow$ *rank-of X* = *card X* $\wedge$ *rank-of X* = *rank-of carrier*
proof (*standard*, *goal-cases LTR RTL*)
case *LTR*
then have *rank-of X* = *card X* **using** *assms indep-iff-rank-of basis-indep* **by** *auto*
moreover have ... = *rank-of carrier*
using *LTR rank-of-eq-card-basis-in*[*of carrier X*] *basis-iff-basis-in* **by** *auto*
ultimately show *?case* **by** *auto*
next
case *RTL*
show *?case*
proof (*rule basisI*)
show *indep X* **using** *assms RTL indep-iff-rank-of* **by** *blast*
next
fix *x*
assume *x*: $x \in$ *carrier* $-$ *X*
show $\neg$ *indep* (*insert x X*)
proof (*rule ccontr*, *goal-cases False*)
case *False*
then have *card* (*insert x X*) $\leq$ *rank-of carrier*
using *assms x indep-inI rank-of-indep-in-le* **by** *auto*
also have ... = *card X* **using** *RTL* **by** *auto*
finally show *?case* **using** *finite-subset*[*OF assms carrier-finite*] *x* **by** *auto*
qed
qed
qed

lemma *circuit-iff-rank-of*:
assumes $X \subseteq$ *carrier*
shows *circuit X* $\longleftrightarrow$ $X \neq \{\}$ $\wedge$ ($\forall x \in X$. *rank-of* ($X - \{x\}$) = *card* ($X - \{x\}$) $\wedge$ *card* ($X - \{x\}$) = *rank-of X*)
proof (*standard*, *goal-cases LTR RTL*)
case *LTR*
then have $X \neq \{\}$ **using** *circuit-nonempty* **by** *auto*
moreover have *indep-remove*: $\bigwedge x.\ x \in X \Longrightarrow$ *rank-of* ($X - \{x\}$) = *card* ($X - \{x\}$)

proof −
fix x
assume $x \in X$
then have *indep* $(X - \{x\})$ **using** *circuit-min-dep*[*OF LTR*] **by** *auto*
moreover have $X - \{x\} \subseteq$ *carrier* **using** *assms* **by** *auto*
ultimately show *rank-of* $(X - \{x\}) =$ *card* $(X - \{x\})$ **using** *indep-iff-rank-of* **by** *auto*
qed
moreover have $\bigwedge x.\ x \in X \Longrightarrow$ *rank-of* $(X - \{x\}) =$ *rank-of* X
proof −
fix x
assume $*$: $x \in X$
have *rank-of* $X \leq$ *card* X **using** *assms rank-of-le* **by** *auto*
moreover have *rank-of* $X \neq$ *card* X **using** *assms LTR circuitD indep-iff-rank-of*[*of* X] **by** *auto*
ultimately have *rank-of* $X <$ *card* X **by** *auto*
then have *rank-of* $X \leq$ *card* $(X - \{x\})$ **using** $*$ *finite-subset*[*OF assms*] *carrier-finite* **by** *auto*
also have $\ldots =$ *rank-of* $(X - \{x\})$ **using** *indep-remove* ‹$x \in X$› **by** *auto*
finally show *rank-of* $(X - \{x\}) =$ *rank-of* X **using** *assms rank-of-mono*[*of* $X - \{x\}$ X] **by** *auto*
qed
ultimately show *?case* **by** *auto*
next
case *RTL*
then have $X \neq \{\}$
and *indep-remove*: $\bigwedge x.\ x \in X \Longrightarrow$ *rank-of* $(X - \{x\}) =$ *card* $(X - \{x\})$
and *dep*: $\bigwedge x.\ x \in X \Longrightarrow$ *rank-of* $(X - \{x\}) =$ *rank-of* X
by *auto*
show *?case* **using** *assms*
proof (*rule circuitI*)
obtain x **where** x: $x \in X$ **using** ‹$X \neq \{\}$› **by** *auto*
then have *rank-of* $X =$ *card* $(X - \{x\})$ **using** *dep indep-remove* **by** *auto*
also have $\ldots <$ *card* X **using** *card-Diff1-less*[*OF finite-subset*[*OF assms carrier-finite*] x] **.**
finally show $\neg$ *indep* X **using** *indep-iff-rank-of*[*OF assms*] **by** *auto*
next
fix x
assume $x \in X$
then show *indep* $(X - \{x\})$ **using** *assms indep-remove*[*of* x] *indep-iff-rank-of*[*of* $X - \{x\}$]
by *auto*
qed
qed

context
fixes $\mathcal{E}$
assumes $*$: $\mathcal{E} \subseteq$ *carrier*
begin

interpretation $\mathcal{E}$: *matroid* $\mathcal{E}$ *indep-in* $\mathcal{E}$
 using $*$ **by** *auto*

lemma *indep-in-iff-rank-of*:
 assumes $X \subseteq \mathcal{E}$
 shows *indep-in* $\mathcal{E}$ $X \longleftrightarrow$ *rank-of* X = *card* X
 using *assms* $\mathcal{E}$.*indep-iff-rank-of* *rank-of-sub-cong*[*OF* $*$ *assms*] **by** *auto*

lemma *basis-in-iff-rank-of*:
 assumes $X \subseteq \mathcal{E}$
 shows *basis-in* $\mathcal{E}$ $X \longleftrightarrow$ *rank-of* X = *card* $X \wedge$ *rank-of* X = *rank-of* $\mathcal{E}$
 using $\mathcal{E}$.*basis-iff-rank-of*[*OF assms*] *rank-of-sub-cong*[*OF* $*$] *assms*
 unfolding *basis-in-def* **by** *auto*

lemma *circuit-in-iff-rank-of*:
 assumes $X \subseteq \mathcal{E}$
 shows *circuit-in* $\mathcal{E}$ $X \longleftrightarrow X \neq \{\} \wedge (\forall x \in X.$ *rank-of* $(X - \{x\})$ = *card* $(X - \{x\}) \wedge$ *card* $(X - \{x\})$ = *rank-of* $X)$
proof –
 have *circuit-in* $\mathcal{E}$ $X \longleftrightarrow \mathcal{E}$.*circuit* X **unfolding** *circuit-in-def* **..**
 also have ... $\longleftrightarrow X \neq \{\} \wedge (\forall x \in X.\ \mathcal{E}$.*rank-of* $(X - \{x\})$ = *card* $(X - \{x\}) \wedge$ *card* $(X - \{x\}) = \mathcal{E}$.*rank-of* $X)$
  using $\mathcal{E}$.*circuit-iff-rank-of*[*OF assms*] **.**
 also have ... $\longleftrightarrow X \neq \{\} \wedge (\forall x \in X.$ *rank-of* $(X - \{x\})$ = *card* $(X - \{x\}) \wedge$ *card* $(X - \{x\})$ = *rank-of* $X)$
 proof –
  {
   fix x
   have $\mathcal{E}$.*rank-of* $(X - \{x\})$ = *rank-of* $(X - \{x\})$ $\mathcal{E}$.*rank-of* X = *rank-of* X
    using *assms* *rank-of-sub-cong*[*OF* $*$, *of* $X - \{x\}$] *rank-of-sub-cong*[*OF* $*$, *of* X] **by** *auto*
    then have $\mathcal{E}$.*rank-of* $(X - \{x\})$ = *card* $(X - \{x\}) \wedge$ *card* $(X - \{x\}) = \mathcal{E}$.*rank-of* $X \longleftrightarrow$ *rank-of* $(X - \{x\})$ = *card* $(X - \{x\}) \wedge$ *card* $(X - \{x\})$ = *rank-of* X
    by *auto*
  }
  then show *?thesis*
   by (*auto simp*: *simp del*: *card-Diff-insert*)
 qed
 finally show *?thesis* **.**
qed

end

2.5 Closure

definition *cl* :: *'a set* $\Rightarrow$ *'a set* **where**
 cl $X \equiv \{x \in$ *carrier*. *rank-of* (*insert* x X) = *rank-of* $X\}$

lemma *clI*:
  assumes $x \in$ *carrier*
  assumes *rank-of* (*insert x X*) = *rank-of X*
  shows $x \in$ *cl X*
  unfolding *cl-def* **using** *assms* **by** *auto*

lemma *cl-altdef*:
  assumes $X \subseteq$ *carrier*
  shows *cl X* = $\bigcup$ {*Y* ∈ *Pow carrier*. *X* ⊆ *Y* ∧ *rank-of Y* = *rank-of X*}
proof −
  {
    fix *x*
    assume ∗: $x \in$ *cl X*
    have $x \in \bigcup$ {*Y* ∈ *Pow carrier*. *X* ⊆ *Y* ∧ *rank-of Y* = *rank-of X*}
    proof
      show *insert x X* ∈ {*Y* ∈ *Pow carrier*. *X* ⊆ *Y* ∧ *rank-of Y* = *rank-of X*}
        using *assms* ∗ **unfolding** *cl-def* **by** *auto*
    qed *auto*
  }
  moreover {
    fix *x*
    assume ∗: $x \in \bigcup$ {*Y* ∈ *Pow carrier*. *X* ⊆ *Y* ∧ *rank-of Y* = *rank-of X*}
    then obtain *Y* **where** *Y*: *x* ∈ *Y* *Y* ⊆ *carrier X* ⊆ *Y rank-of Y* = *rank-of X*
by *auto*
    have *rank-of* (*insert x X*) = *rank-of X*
    proof −
      have *rank-of* (*insert x X*) ≤ *rank-of X*
      proof −
        have *insert x X* ⊆ *Y* **using** *Y* **by** *auto*
        then show *?thesis* **using** *rank-of-mono*[*of insert x X Y*] *Y* **by** *auto*
      qed
      moreover have *rank-of X* ≤ *rank-of* (*insert x X*) **using** *Y* **by** (*auto intro*:
rank-of-mono)
      ultimately show *?thesis* **by** *auto*
    qed
    then have $x \in$ *cl X* **using** ∗ **unfolding** *cl-def* **by** *auto*
  }
  ultimately show *?thesis* **by** *blast*
qed

lemma *cl-rank-of*: $x \in$ *cl X* ⟹ *rank-of* (*insert x X*) = *rank-of X*
  unfolding *cl-def* **by** *auto*

lemma *cl-subset-carrier*: *cl X* ⊆ *carrier*
  unfolding *cl-def* **by** *auto*

lemmas *clD* = *cl-rank-of cl-subset-carrier*

lemma *cl-subset*:
  assumes $X \subseteq$ *carrier*
  shows $X \subseteq$ *cl* X
  using *assms* **using** *insert-absorb*[*of* - X] **by** (*auto intro*!: *clI*)

lemma *cl-mono*:
  assumes $X \subseteq Y$
  assumes $Y \subseteq$ *carrier*
  shows *cl* $X \subseteq$ *cl* Y
proof
  fix x
  assume $x \in$ *cl* X
  then have $x \in$ *carrier* **using** *cl-subset-carrier* **by** *auto*

  have *insert* x $X \subseteq$ *carrier*
    using *assms* ‹$x \in$ *cl* X› *cl-subset-carrier*[*of* X] **by** *auto*
  then interpret *X-insert*: *matroid insert* x X *indep-in* (*insert* x X) **by** *auto*

  have *insert* x $Y \subseteq$ *carrier*
    using *assms* ‹$x \in$ *cl* X› *cl-subset-carrier*[*of* X] **by** *auto*
  then interpret *Y-insert*: *matroid insert* x Y *indep-in* (*insert* x Y) **by** *auto*

  show $x \in$ *cl* Y **using** ‹$x \in$ *carrier*›
  proof (*rule clI*, *cases* $x \in X$)
    case *True*
    then show *rank-of* (*insert* x Y) = *rank-of* Y **using** *assms insert-absorb*[*of* x
Y] **by** *auto*
  next
    case *False*
    obtain B_X **where** B_X: *basis-in* X B_X **using** *assms basis-in-ex*[*of* X] **by** *auto*

    have *basis-in* (*insert* x X) B_X
    proof −
      have *rank-of* B_X = *card* B_X $\wedge$ *rank-of* B_X = *rank-of* (*insert* x X)
      proof −
        have *rank-of* B_X = *card* B_X $\wedge$ *rank-of* B_X = *rank-of* X
          using *assms* B_X
            basis-in-subset-carrier[**where** $\mathcal{E} = X$ **and** $X = B_X$]
            basis-in-iff-rank-of[**where** $\mathcal{E} = X$ **and** $X = B_X$]
          by *blast*
        then show *?thesis* **using** *cl-rank-of*[*OF* ‹$x \in$ *cl* X›] **by** *auto*
      qed
      then show *?thesis*
      using *assms basis-in-subset-carrier*[*OF* - B_X] ‹$x \in$ *carrier*› *basis-in-iff-rank-of*[*of*
insert x X B_X]
        by *auto*
    qed

have *indep-in* (*insert x Y*) B_X
using *assms basis-in-indep-in*[*OF* - B_X] *indep-in-subI-subset*[*of X insert x Y*]
by *auto*
then obtain B_Y **where** B_Y: *basis-in* (*insert x Y*) B_Y $B_X \subseteq B_Y$
using *assms* ‹$x \in$ *carrier*› *indep-in-iff-subset-basis-in*[*of insert x Y* B_X] **by**
auto

have *basis-in Y* B_Y
proof −
have $x \notin B_Y$
proof (*rule ccontr*, *goal-cases False*)
case *False*
then have *insert x* $B_X \subseteq B_Y$ **using** ‹$B_X \subseteq B_Y$› **by** *auto*
then have *indep-in* (*insert x Y*) (*insert x* B_X)
using *assms* ‹$x \in$ *carrier*›
B_Y *basis-in-indep-in*[**where** $\mathcal{E}$ = *insert x Y* **and** $X = B_Y$]
indep-in-subset[**where** $\mathcal{E}$ = *insert x Y* **and** $X = B_Y$ **and** Y = *insert x*
B_X]
by *auto*
then have *indep-in* (*insert x X*) (*insert x* B_X)
using *assms* B_X
basis-in-subset-carrier[**where** $\mathcal{E} = X$ **and** $X = B_X$]
indep-in-supI[**where** $\mathcal{E}$ = *insert x Y* **and** $\mathcal{E}'$ = *insert x X* **and** X =
insert x B_X]
by *auto*
moreover have $x \in$ *insert x X* $- B_X$
using *assms* ‹$x \notin X$› B_X *basis-in-subset-carrier*[**where** $\mathcal{E} = X$ **and** X =
B_X] **by** *auto*
ultimately show *?case*
using *assms* ‹$x \in$ *carrier*› ‹*basis-in* (*insert x X*) B_X›
basis-in-max-indep-in[**where** $\mathcal{E}$ = *insert x X* **and** $X = B_X$ **and** $x = x$]
by *auto*
qed
then show *?thesis*
using *assms* ‹$x \in$ *carrier*› B_Y *basis-in-subset-carrier*[*of insert x Y* B_Y]
basis-in-supI[**where** $\mathcal{E}$ = *insert x Y* **and** $\mathcal{E}' = Y$ **and** $B = B_Y$] **by** *auto*
qed

show *rank-of* (*insert x Y*) = *rank-of Y*
proof −
have *rank-of* (*insert x Y*) = *card* B_Y
using *assms* ‹$x \in$ *carrier*› ‹*basis-in* (*insert x Y*) B_Y› *basis-in-subset-carrier*
using *basis-in-iff-rank-of*[**where** $\mathcal{E}$ = *insert x Y* **and** $X = B_Y$]
by *auto*
also have ... = *rank-of Y*
using *assms* ‹$x \in$ *carrier*› ‹*basis-in Y* B_Y› *basis-in-subset-carrier*
using *basis-in-iff-rank-of*[**where** $\mathcal{E} = Y$ **and** $X = B_Y$]
by *auto*
finally show *?thesis* **.**

```
    qed
  qed
qed

lemma cl-insert-absorb:
  assumes X ⊆ carrier
  assumes x ∈ cl X
  shows cl (insert x X) = cl X
proof
  show cl (insert x X) ⊆ cl X
  proof (standard, goal-cases elem)
    case (elem y)
    then have *: x ∈ carrier y ∈ carrier using assms cl-subset-carrier by auto

    have rank-of (insert y X) = rank-of (insert y (insert x X))
    proof −
      have rank-of (insert y X) ≤ rank-of (insert y (insert x X))
        using assms * by (auto intro: rank-of-mono)
      moreover have rank-of (insert y (insert x X)) = rank-of (insert y X)
      proof −
        have insert y (insert x X) = insert x (insert y X) by auto
       then have rank-of (insert y (insert x X)) = rank-of (insert x (insert y X))
by auto
        also have ... = rank-of (insert y X)
        proof −
          have cl X ⊆ cl (insert y X) by (rule cl-mono) (auto simp add: assms ‹y
∈ carrier›)
          then have x ∈ cl (insert y X) using assms by auto
          then show ?thesis unfolding cl-def by auto
        qed
        finally show ?thesis .
      qed
      ultimately show ?thesis by auto
    qed
    also have ... = rank-of (insert x X) using elem using cl-rank-of by auto
    also have ... = rank-of X using assms cl-rank-of by auto
    finally show y ∈ cl X using * by (auto intro: clI)
  qed
next
  have insert x X ⊆ carrier using assms cl-subset-carrier by auto
  moreover have X ⊆ insert x X using assms by auto
  ultimately show cl X ⊆ cl (insert x X) using assms cl-subset-carrier cl-mono
by auto
qed

lemma cl-cl-absorb:
  assumes X ⊆ carrier
  shows cl (cl X) = cl X
proof
```

```
  show cl (cl X) ⊆ cl X
  proof (standard, goal-cases elem)
    case (elem x)
    then have x ∈ carrier using cl-subset-carrier by auto
    then show ?case
    proof (rule clI)
      have rank-of (insert x X) ≥ rank-of X
        using assms ‹x ∈ carrier› rank-of-mono[of X insert x X] by auto
      moreover have rank-of (insert x X) ≤ rank-of X
      proof −
        have rank-of (insert x X) ≤ rank-of (insert x (cl X))
          using assms ‹x ∈ carrier› cl-subset-carrier cl-subset[of X]
                rank-of-mono[of insert x X insert x (cl X)] by auto
        also have ... = rank-of (cl X) using elem cl-rank-of by auto
        also have ... = rank-of (X ∪ (cl X − X))
          using Diff-partition[OF cl-subset[OF assms]] by auto
        also have ... = rank-of X using ‹X ⊆ carrier›
        proof (rule rank-of-Un-absorbI)
          show cl X − X ⊆ carrier using assms cl-subset-carrier by auto
        next
          fix y
          assume y ∈ cl X − X − X
          then show rank-of (insert y X) = rank-of X unfolding cl-def by auto
        qed
        finally show ?thesis .
      qed
      ultimately show rank-of (insert x X) = rank-of X by auto
    qed
  qed
next
  show cl X ⊆ cl (cl X) using cl-subset[OF cl-subset-carrier] by auto
qed

lemma cl-augment:
  assumes X ⊆ carrier
  assumes x ∈ carrier
  assumes y ∈ cl (insert x X) − cl X
  shows x ∈ cl (insert y X)
  using ‹x ∈ carrier›
proof (rule clI)
  have rank-of (insert y X) ≤ rank-of (insert x (insert y X))
    using assms cl-subset-carrier by (auto intro: rank-of-mono)
  moreover have rank-of (insert x (insert y X)) ≤ rank-of (insert y X)
  proof −
    have rank-of (insert x (insert y X)) = rank-of (insert y (insert x X))
    proof −
      have insert x (insert y X) = insert y (insert x X) by auto
      then show ?thesis by auto
    qed
```

also have *rank-of* (*insert y* (*insert x X*)) = *rank-of* (*insert x X*)
  using *assms cl-def* **by** *auto*
also have ... ≤ *Suc* (*rank-of X*)
  using *assms cl-subset-carrier* **by** (*auto intro*: *rank-of-insert-le*)
also have ... = *rank-of* (*insert y X*)
proof −
  have *rank-of* (*insert y X*) ≤ *Suc* (*rank-of X*)
    using *assms cl-subset-carrier* **by** (*auto intro*: *rank-of-insert-le*)
  moreover have *rank-of* (*insert y X*) ≠ *rank-of X*
    using *assms cl-def* **by** *auto*
  moreover have *rank-of X* ≤ *rank-of* (*insert y X*)
    using *assms cl-subset-carrier* **by** (*auto intro*: *rank-of-mono*)
  ultimately show *?thesis* **by** *auto*
qed
finally show *?thesis* **.**
qed
ultimately show *rank-of* (*insert x* (*insert y X*)) = *rank-of* (*insert y X*) **by** *auto*
qed

lemma *clI-insert*:
assumes *x* ∈ *carrier*
assumes *indep X*
assumes ¬ *indep* (*insert x X*)
shows *x* ∈ *cl X*
using ‹*x* ∈ *carrier*›
proof (*rule clI*)
have ∗: *X* ⊆ *carrier* **using** *assms indep-subset-carrier* **by** *auto*
then have ∗∗: *insert x X* ⊆ *carrier* **using** *assms* **by** *auto*

have *indep-in* (*insert x X*) *X* **using** *assms* **by** (*auto intro*: *indep-inI*)
then obtain *B* **where** *B*: *basis-in* (*insert x X*) *B X* ⊆ *B*
  using *assms indep-in-iff-subset-basis-in*[*OF* ∗∗] **by** *auto*
have *x* ∉ *B*
proof (*rule ccontr*, *goal-cases False*)
  case *False*
  then have *indep-in* (*insert x X*) (*insert x X*)
    using *B indep-in-subset*[*OF* ∗∗ *basis-in-indep-in*[*OF* ∗∗]] **by** *auto*
  then show *?case* **using** *assms indep-in-indep* **by** *auto*
qed

have *basis-in X B* **using** ∗
proof (*rule basis-inI*, *goal-cases indep max-indep*)
  case *indep*
  show *?case*
  proof (*rule indep-in-supI*[**where** $\mathcal{E}$ = *insert x X*])
    show *B* ⊆ *X* **using** *B basis-in-subset-carrier*[*OF* ∗∗] ‹*x* ∉ *B*› **by** *auto*
  next
    show *indep-in* (*insert x X*) *B* **using** *basis-in-indep-in*[*OF* ∗∗ *B*(*1*)] **.**
  qed *auto*

```
  next
    case (max-indep y)
    then have ¬ indep-in (insert x X) (insert y B)
      using B basis-in-max-indep-in[OF **] by auto
    then show ?case by (auto intro: indep-in-subI-subset)
  qed
  then show rank-of (insert x X) = rank-of X
   using B rank-of-eq-card-basis-in[OF *] rank-of-eq-card-basis-in[OF **] by auto
qed

lemma indep-in-carrier [simp]: indep-in carrier = indep
  using indep-subset-carrier by (auto simp: indep-in-def fun-eq-iff)

context
  fixes I
  defines I ≡ (λX. X ⊆ carrier ∧ (∀x∈X. x ∉ cl (X − {x})))
begin

lemma I-mono: I Y if Y ⊆ X I X for X Y :: 'a set
proof −
  have ∀x∈Y. x ∉ cl (Y − {x})
  proof (intro ballI)
    fix x assume x: x ∈ Y
    with that have cl (Y − {x}) ⊆ cl (X − {x})
      by (intro cl-mono) (auto simp: I-def)
    with that and x show x ∉ cl (Y − {x}) by (auto simp: I-def)
  qed
  with that show ?thesis by (auto simp: I-def)
qed

lemma clI':
  assumes I X x ∈ carrier ¬I (insert x X)
  shows   x ∈ cl X
proof −
  from assms have x: x ∉ X by (auto simp: insert-absorb)
  from assms obtain y where y: y ∈ insert x X y ∈ cl (insert x X − {y})
    by (force simp: I-def)
  show x ∈ cl X
  proof (cases x = y)
    case True
    thus ?thesis using assms x y by (auto simp: I-def)
  next
    case False
    have y ∈ cl (insert x X − {y}) by fact
    also from False have insert x X − {y} = insert x (X − {y}) by auto
    finally have y ∈ cl (insert x (X − {y})) − cl (X − {y})
      using assms False y unfolding I-def by blast
    hence x ∈ cl (insert y (X − {y}))
      using cl-augment[of X − {y} x y] assms False y by (auto simp: I-def)
```

also from y **and** *False* **have** $insert\ y\ (X - \{y\}) = X$ **by** *auto*
finally show *?thesis* **.**
qed
qed

lemma *matroid-I*: *matroid carrier I*
proof (*unfold-locales*, *goal-cases*)
show *finite carrier* **by** (*rule carrier-finite*)
next
case ($4\ X\ Y$)
have $\forall x{\in}Y.\ x \notin cl\ (Y - \{x\})$
proof (*intro ballI*)
fix x **assume** x: $x \in Y$
with 4 **have** $cl\ (Y - \{x\}) \subseteq cl\ (X - \{x\})$
by (*intro cl-mono*) (*auto simp*: *I-def*)
with 4 **and** x **show** $x \notin cl\ (Y - \{x\})$ **by** (*auto simp*: *I-def*)
qed
with 4 **show** *?case* **by** (*auto simp*: *I-def*)
next
case ($5\ X\ Y$)
have $\sim(\exists X\ Y.\ I\ X \wedge I\ Y \wedge card\ X < card\ Y \wedge (\forall x{\in}Y{-}X.\ \neg I\ (insert\ x\ X)))$
proof
assume $*$: $\exists X\ Y.\ I\ X \wedge I\ Y \wedge card\ X < card\ Y \wedge (\forall x{\in}Y{-}X.\ \neg I\ (insert\ x\ X))$ (**is** $\exists X\ Y.\ ?P\ X\ Y$)
define n **where** $n = Max\ ((\lambda(X,\ Y).\ card\ (X \cap Y))\ `\ \{(X,\ Y).\ ?P\ X\ Y\})$
have $\{(X,\ Y).\ ?P\ X\ Y\} \subseteq Pow\ carrier \times Pow\ carrier$
by (*auto simp*: *I-def*)
hence *finite*: *finite* $\{(X,\ Y).\ ?P\ X\ Y\}$
by (*rule finite-subset*) (*insert carrier-finite*, *auto*)
hence $n \in ((\lambda(X,\ Y).\ card\ (X \cap Y))\ `\ \{(X,\ Y).\ ?P\ X\ Y\})$
unfolding *n-def* **using** $*$ **by** (*intro Max-in finite-imageI*) *auto*
then obtain $X\ Y$ **where** XY: $?P\ X\ Y\ n = card\ (X \cap Y)$
by *auto*
hence *finite′*: *finite X finite Y*
using *finite-subset*[*OF - carrier-finite*] XY **by** (*auto simp*: *I-def*)
from XY *finite′* **have** $\sim(Y \subseteq X)$
using *card-mono*[*of X Y*] **by** *auto*
then obtain y **where** y: $y \in Y - X$ **by** *blast*

have *False*
proof (*cases* $X \subseteq cl\ (Y - \{y\})$)
case *True*
from y XY **have** [*simp*]: $y \in carrier$ **by** (*auto simp*: *I-def*)
assume $X \subseteq cl\ (Y - \{y\})$
hence $cl\ X \subseteq cl\ (cl\ (Y - \{y\}))$
by (*intro cl-mono cl-subset-carrier*)
also have $\ldots = cl\ (Y - \{y\})$
using XY **by** (*intro cl-cl-absorb*) (*auto simp*: *I-def*)

```
      finally have cl X ⊆ cl (Y − {y}) .
      moreover have y ∉ cl (Y − {y})
        using y I-def XY(1) by blast
      ultimately have y ∉ cl X by blast
      thus False unfolding I-def
        using XY y clI' ‹y ∈ carrier› by blast
    next
      case False
      with y XY have [simp]: y ∈ carrier by (auto simp: I-def)
      assume ¬(X ⊆ cl (Y − {y}))
      then obtain t where t: t ∈ X t ∉ cl (Y − {y})
        by auto
      with XY have [simp]: t ∈ carrier by (auto simp: I-def)

      have t ∈ X − Y
        using t y clI[of t Y − {y}] by (cases t = y) (auto simp: insert-absorb)
      moreover have I (Y − {y}) using XY(1) I-mono[of Y − {y} Y] by blast
      ultimately have ∗: I (insert t (Y − {y}))
        using clI'[of Y − {y} t] t by auto

      from XY have finite Y
        by (intro finite-subset[OF - carrier-finite]) (auto simp: I-def)
      moreover from y have Y ≠ {} by auto
      ultimately have [simp]: card (insert t (Y − {y})) = card Y using ‹t ∈ X
− Y› y
        by (simp add: Suc-diff-Suc card-gt-0-iff)

      have ∃ x∈Y − X. I (insert x X)
      proof (rule ccontr)
        assume ¬ ?thesis
        hence ?P X (insert t (Y − {y})) using XY ∗ ‹t ∈ X − Y›
          by auto
        hence card (X ∩ insert t (Y − {y})) ≤ n
          unfolding n-def using finite by (intro Max-ge) auto
        also have X ∩ insert t (Y − {y}) = insert t ((X ∩ Y) − {y})
          using y ‹t ∈ X − Y› by blast
        also have card ... = Suc (card (X ∩ Y))
          using y ‹t ∈ X − Y› ‹finite Y› by (simp add: card.insert-remove)
        finally show False using XY by simp
      qed
      with XY show False by blast
    qed
    thus False .
  qed
  with 5 show ?case by auto
qed (auto simp: I-def)

end
```

```
definition cl-in where cl-in ℰ X = matroid.cl ℰ (indep-in ℰ) X

lemma cl-eq-cl-in:
  assumes X ⊆ carrier
  shows cl X = cl-in carrier X
proof −
  interpret ℰ: matroid carrier indep-in carrier
    by (intro matroid-subset) auto
  have cl X = {x ∈ carrier. rank-of (insert x X) = rank-of X}
    unfolding cl-def by auto
  also have ... = {x ∈ carrier. ℰ.rank-of (insert x X) = ℰ.rank-of X}
    using rank-of-sub-cong[of carrier] assms by auto
  also have ... = cl-in carrier X
    unfolding cl-in-def ℰ.cl-def by auto
  finally show ?thesis .
qed

context
  fixes ℰ
  assumes ∗: ℰ ⊆ carrier
begin

interpretation ℰ: matroid ℰ indep-in ℰ
  using ∗ by auto

lemma cl-inI-aux: x ∈ ℰ.cl X ⟹ x ∈ cl-in ℰ X
  unfolding cl-in-def by auto

lemma cl-inD-aux: x ∈ cl-in ℰ X ⟹ x ∈ ℰ.cl X
  unfolding cl-in-def by auto

lemma cl-inI:
  assumes X ⊆ ℰ
  assumes x ∈ ℰ
  assumes rank-of (insert x X) = rank-of X
  shows x ∈ cl-in ℰ X
proof −
  have ℰ.rank-of (insert x X) = rank-of (insert x X) ℰ.rank-of X = rank-of X
    using assms rank-of-sub-cong[OF ∗] by auto
  then show ?thesis unfolding cl-in-def using assms by (auto intro: ℰ.clI)
qed

lemma cl-in-altdef:
  assumes X ⊆ ℰ
  shows cl-in ℰ X = ⋃{Y ∈ Pow ℰ. X ⊆ Y ∧ rank-of Y = rank-of X}
  unfolding cl-in-def
proof (safe, goal-cases LTR RTL)
  case (LTR x)
  then have x ∈ ⋃{Y ∈ Pow ℰ. X ⊆ Y ∧ ℰ.rank-of Y = ℰ.rank-of X}
```

using $\mathcal{E}$.*cl-altdef*[*OF assms*] **by** *auto*
then obtain Y **where** Y: $x \in Y$ $Y \in Pow\ \mathcal{E}$ $X \subseteq Y$ $\mathcal{E}$.*rank-of* Y = $\mathcal{E}$.*rank-of* X **by** *auto*
then show *?case* **using** *rank-of-sub-cong*[*OF* ∗] **by** *auto*
next
case (*RTL x Y*)
then have $x \in \bigcup\{Y \in Pow\ \mathcal{E}.\ X \subseteq Y \wedge \mathcal{E}.rank\text{-}of\ Y = \mathcal{E}.rank\text{-}of\ X\}$
using *rank-of-sub-cong*[*OF* ∗, *of X*] *rank-of-sub-cong*[*OF* ∗, *of Y*] **by** *auto*
then show *?case* **using** $\mathcal{E}$.*cl-altdef*[*OF assms*] **by** *auto*
qed

lemma *cl-in-subset-carrier*: *cl-in* $\mathcal{E}$ $X \subseteq \mathcal{E}$
using $\mathcal{E}$.*cl-subset-carrier* **unfolding** *cl-in-def* **.**

lemma *cl-in-rank-of*:
assumes $X \subseteq \mathcal{E}$
assumes $x \in$ *cl-in* $\mathcal{E}$ X
shows *rank-of* (*insert x X*) = *rank-of X*
proof −
have $\mathcal{E}$.*rank-of* (*insert x X*) = $\mathcal{E}$.*rank-of X*
using *assms* $\mathcal{E}$.*cl-rank-of* **unfolding** *cl-in-def* **by** *auto*
moreover have $\mathcal{E}$.*rank-of* (*insert x X*) = *rank-of* (*insert x X*)
using *assms rank-of-sub-cong*[*OF* ∗, *of insert x X*] *cl-in-subset-carrier* **by** *auto*
moreover have $\mathcal{E}$.*rank-of X* = *rank-of X*
using *assms rank-of-sub-cong*[*OF* ∗] **by** *auto*
ultimately show *?thesis* **by** *auto*
qed

lemmas *cl-inD* = *cl-in-rank-of cl-in-subset-carrier*

lemma *cl-in-subset*:
assumes $X \subseteq \mathcal{E}$
shows $X \subseteq$ *cl-in* $\mathcal{E}$ X
using $\mathcal{E}$.*cl-subset*[*OF assms*] **unfolding** *cl-in-def* **.**

lemma *cl-in-mono*:
assumes $X \subseteq Y$
assumes $Y \subseteq \mathcal{E}$
shows *cl-in* $\mathcal{E}$ $X \subseteq$ *cl-in* $\mathcal{E}$ Y
using $\mathcal{E}$.*cl-mono*[*OF assms*] **unfolding** *cl-in-def* **.**

lemma *cl-in-insert-absorb*:
assumes $X \subseteq \mathcal{E}$
assumes $x \in$ *cl-in* $\mathcal{E}$ X
shows *cl-in* $\mathcal{E}$ (*insert x X*) = *cl-in* $\mathcal{E}$ X
using *assms* $\mathcal{E}$.*cl-insert-absorb* **unfolding** *cl-in-def* **by** *auto*

lemma *cl-in-augment*:
assumes $X \subseteq \mathcal{E}$

assumes $x \in \mathcal{E}$
assumes $y \in$ *cl-in* $\mathcal{E}$ (*insert* x X) $-$ *cl-in* $\mathcal{E}$ X
shows $x \in$ *cl-in* $\mathcal{E}$ (*insert* y X)
using *assms* $\mathcal{E}$.*cl-augment* **unfolding** *cl-in-def* **by** *auto*

lemmas *cl-inI-insert* = *cl-inI-aux*[*OF* $\mathcal{E}$.*clI-insert*]

end

lemma *cl-in-subI*:
assumes $X \subseteq \mathcal{E}'$ $\mathcal{E}' \subseteq \mathcal{E}$ $\mathcal{E} \subseteq$ *carrier*
shows *cl-in* $\mathcal{E}'$ $X \subseteq$ *cl-in* $\mathcal{E}$ X
proof (*safe*, *goal-cases elem*)
case (*elem* x)
then have $x \in \mathcal{E}'$ *rank-of* (*insert* x X) = *rank-of* X
using *assms* *cl-inD*[**where** $\mathcal{E} = \mathcal{E}'$ **and** $X = X$] **by** *auto*
then show $x \in$ *cl-in* $\mathcal{E}$ X **using** *assms* **by** (*auto intro*: *cl-inI*)
qed

context
fixes $\mathcal{E}$
assumes $*$: $\mathcal{E} \subseteq$ *carrier*
begin

interpretation $\mathcal{E}$: *matroid* $\mathcal{E}$ *indep-in* $\mathcal{E}$
using $*$ **by** *auto*

lemma *cl-in-sub-cong*:
assumes $X \subseteq \mathcal{E}'$ $\mathcal{E}' \subseteq \mathcal{E}$
shows $\mathcal{E}$.*cl-in* $\mathcal{E}'$ X = *cl-in* $\mathcal{E}'$ X
proof (*safe*, *goal-cases LTR RTL*)
case (*LTR* x)
then have $x \in \mathcal{E}'$ $\mathcal{E}$.*rank-of* (*insert* x X) = $\mathcal{E}$.*rank-of* X
using *assms*
$\mathcal{E}$.*cl-in-rank-of*[**where** $\mathcal{E} = \mathcal{E}'$ **and** $X = X$ **and** $x = x$]
$\mathcal{E}$.*cl-in-subset-carrier*[**where** $\mathcal{E} = \mathcal{E}'$]
by *auto*
moreover have $\mathcal{E}$.*rank-of* X = *rank-of* X
using *assms* *rank-of-sub-cong*[*OF* $*$] **by** *auto*
moreover have $\mathcal{E}$.*rank-of* (*insert* x X) = *rank-of* (*insert* x X)
using *assms* *rank-of-sub-cong*[*OF* $*$, *of insert* x X] ‹$x \in \mathcal{E}'$› **by** *auto*
ultimately show *?case* **using** *assms* $*$ **by** (*auto intro*: *cl-inI*)
next
case (*RTL* x)
then have $x \in \mathcal{E}'$ *rank-of* (*insert* x X) = *rank-of* X
using $*$ *assms* *cl-inD*[**where** $\mathcal{E} = \mathcal{E}'$ **and** $X = X$] **by** *auto*
moreover have $\mathcal{E}$.*rank-of* X = *rank-of* X
using *assms* *rank-of-sub-cong*[*OF* $*$] **by** *auto*
moreover have $\mathcal{E}$.*rank-of* (*insert* x X) = *rank-of* (*insert* x X)

using *assms rank-of-sub-cong*[*OF* $*$, *of insert x X*] ‹$x \in \mathcal{E}$'› **by** *auto*
ultimately show *?case* **using** *assms* **by** (*auto intro*: $\mathcal{E}$.*cl-inI*)
qed

end
end
end

References

[1] J. Oxley. What is a matroid?, 2003.