Relational Minimum Spanning Tree Algorithms

Walter Guttmann and Nicolas Robinson-O'Brien

February 6, 2026

Abstract

We verify the correctness of Prim's, Kruskal's and Borůvka's minimum spanning tree algorithms based on algebras for aggregation and minimisation.

Contents

1 Overview **1**
1.1 Prim's and Kruskal's minimum spanning tree algorithms . . . 2
1.2 Borůvka's minimum spanning tree algorithm 2

2 Kruskal's Minimum Spanning Tree Algorithm **2**

3 Prim's Minimum Spanning Tree Algorithm **15**

4 Borůvka's Minimum Spanning Tree Algorithm **26**
4.1 General results 26
4.2 Forests modulo an equivalence 31
4.3 An operation to select components 40
4.4 m-k-Stone-Kleene relation algebras 40
4.4.1 Components of forests and forests modulo an equivalence 43
4.4.2 Identifying arcs 51
4.4.3 Comparison of edge weights 67
4.4.4 Maintenance of algorithm invariants 75
4.5 Formalization and correctness proof 104

1 Overview

The theories described in this document prove the correctness of Prim's, Kruskal's and Borůvka's minimum spanning tree algorithms. Specifications and algorithms work in Stone-Kleene relation algebras extended by operations for aggregation and minimisation. The algorithms are implemented in a simple imperative language and their proof uses Hoare logic. The correctness proofs are discussed in [3, 5, 6, 8].

1.1 Prim's and Kruskal's minimum spanning tree algorithms

A framework based on Stone relation algebras and Kleene algebras and extended by operations for aggregation and minimisation was presented by the first author in [3, 5] and used to formally verify the correctness of Prim's minimum spanning tree algorithm. It was extended in [6] and applied to prove the correctness of Kruskal's minimum spanning tree algorithm.

Two theories, one each for Prim's and Kruskal's algorithms, prove total correctness of these algorithms. As case studies for the algebraic framework, these two theories combined were originally part of another AFP entry [4].

1.2 Borůvka's minimum spanning tree algorithm

Otakar Borůvka formalised the minimum spanning tree problem and proposed a solution to it [1]. Borůvka's original paper is written in Czech; translations of varying completeness can be found in [2, 7].

The theory for Borůvka's minimum spanning tree algorithm proves partial correctness of this algorithm. This work is based on the same algebraic framework as the proof of Kruskal's algorithm; in particular it uses many theories from the hierarchy underlying [4].

The theory for Borůvka's algorithm formally verifies results from the second author's Master's thesis [8].

2 Kruskal's Minimum Spanning Tree Algorithm

In this theory we prove total correctness of Kruskal's minimum spanning tree algorithm. The proof uses the following steps [6]. We first establish that the algorithm terminates and constructs a spanning tree. This is a constructive proof of the existence of a spanning tree; any spanning tree algorithm could be used for this. We then conclude that a minimum spanning tree exists. This is necessary to establish the invariant for the actual correctness proof, which shows that Kruskal's algorithm produces a minimum spanning tree.

theory *Kruskal*

imports *HOL−Hoare.Hoare-Logic Aggregation-Algebras.Aggregation-Algebras*

begin

context *m-kleene-algebra*
begin

definition *spanning-forest f g* ≡ *forest f* ∧ *f* ≤ −−*g* ∧ *components g* ≤ *forest-components f* ∧ *regular f*
definition *minimum-spanning-forest f g* ≡ *spanning-forest f g* ∧ (∀ *u* . *spanning-forest u g* ⟶ *sum* (*f* ⊓ *g*) ≤ *sum* (*u* ⊓ *g*))

definition *kruskal-spanning-invariant f g h* ≡ *symmetric g* ∧ *h* = h^T ∧ *g* ⊓ −−*h* = *h* ∧ *spanning-forest f* (−*h* ⊓ *g*)
definition *kruskal-invariant f g h* ≡ *kruskal-spanning-invariant f g h* ∧ (∃ *w* . *minimum-spanning-forest w g* ∧ *f* ≤ *w* ⊔ w^T)

We first show two verification conditions which are used in both correctness proofs.

lemma *kruskal-vc-1*:
assumes *symmetric g*
shows *kruskal-spanning-invariant bot g g*
proof (*unfold kruskal-spanning-invariant-def*, *intro conjI*)
show *symmetric g*
using *assms* **by** *simp*
next
show *g* = g^T
using *assms* **by** *simp*
next
show *g* ⊓ −−*g* = *g*
using *inf.sup-monoid.add-commute selection-closed-id* **by** *simp*
next
show *spanning-forest bot* (−*g* ⊓ *g*)
using *star.circ-transitive-equal spanning-forest-def* **by** *simp*
qed

lemma *kruskal-vc-2*:
assumes *kruskal-spanning-invariant f g h*
and *h* ≠ *bot*
shows (*minarc h* ≤ −*forest-components f* ⟶ *kruskal-spanning-invariant* ((*f* ⊓ −(*top* ∗ *minarc h* ∗ $f^{T\star}$)) ⊔ (*f* ⊓ *top* ∗ *minarc h* ∗ $f^{T\star}$)^T ⊔ *minarc h*) *g* (*h* ⊓ −*minarc h* ⊓ −*minarc* h^T)
∧ *card* { *x* . *regular x* ∧ *x* ≤ −−*h* ∧ *x* ≤ −*minarc h* ∧ *x* ≤ −*minarc* h^T } < *card* { *x* . *regular x* ∧ *x* ≤ −−*h* }) ∧
(¬ *minarc h* ≤ −*forest-components f* ⟶ *kruskal-spanning-invariant f g* (*h* ⊓ −*minarc h* ⊓ −*minarc* h^T)
∧ *card* { *x* . *regular x* ∧ *x* ≤ −−*h* ∧ *x* ≤ −*minarc h* ∧ *x* ≤ −*minarc* h^T } < *card* { *x* . *regular x* ∧ *x* ≤ −−*h* })
proof −
let *?e* = *minarc h*
let *?f* = (*f* ⊓ −(*top* ∗ *?e* ∗ $f^{T\star}$)) ⊔ (*f* ⊓ *top* ∗ *?e* ∗ $f^{T\star}$)^T ⊔ *?e*
let *?h* = *h* ⊓ −*?e* ⊓ −$?e^T$
let *?F* = *forest-components f*
let *?n1* = *card* { *x* . *regular x* ∧ *x* ≤ −−*h* }
let *?n2* = *card* { *x* . *regular x* ∧ *x* ≤ −−*h* ∧ *x* ≤ −*?e* ∧ *x* ≤ −$?e^T$ }
have *1*: *regular f* ∧ *regular ?e*
by (*metis assms*(*1*) *kruskal-spanning-invariant-def spanning-forest-def minarc-regular*)
hence *2*: *regular ?f* ∧ *regular ?F* ∧ *regular* ($?e^T$)
using *regular-closed-star regular-conv-closed regular-mult-closed* **by** *simp*
have *3*: ¬ *?e* ≤ −*?e*

using *assms(2) inf.orderE minarc-bot-iff* **by** *fastforce*
have *4*: *?n2 < ?n1*
apply (*rule psubset-card-mono*)
using *finite-regular* **apply** *simp*
using *1 3 kruskal-spanning-invariant-def minarc-below* **by** *auto*
show (*?e ≤ −?F ⟶ kruskal-spanning-invariant ?f g ?h ∧ ?n2 < ?n1*) ∧ (¬ *?e ≤ −?F ⟶ kruskal-spanning-invariant f g ?h ∧ ?n2 < ?n1*)
proof (*rule conjI*)
have *5*: *injective ?f*
apply (*rule kruskal-injective-inv*)
using *assms(1) kruskal-spanning-invariant-def spanning-forest-def* **apply** *simp*
apply (*simp add*: *covector-mult-closed*)
apply (*simp add*: *comp-associative comp-isotone star.right-plus-below-circ*)
apply (*meson mult-left-isotone order-lesseq-imp star-outer-increasing top.extremum*)
using *assms(1,2) kruskal-spanning-invariant-def kruskal-injective-inv-2 minarc-arc spanning-forest-def* **apply** *simp*
using *assms(2) arc-injective minarc-arc* **apply** *blast*
using *assms(1,2) kruskal-spanning-invariant-def kruskal-injective-inv-3 minarc-arc spanning-forest-def* **by** *simp*
show *?e ≤ −?F ⟶ kruskal-spanning-invariant ?f g ?h ∧ ?n2 < ?n1*
proof
assume *6*: *?e ≤ −?F*
have *7*: *equivalence ?F*
using *assms(1) kruskal-spanning-invariant-def forest-components-equivalence spanning-forest-def* **by** *simp*
have $?e^T * top * ?e^T = ?e^T$
using *assms(2)* **by** (*simp add*: *arc-top-arc minarc-arc*)
hence $?e^T * top * ?e^T \leq -?F$
using *6 7 conv-complement conv-isotone* **by** *fastforce*
hence *8*: $?e * ?F * ?e = bot$
using *le-bot triple-schroeder-p* **by** *simp*
show *kruskal-spanning-invariant ?f g ?h ∧ ?n2 < ?n1*
proof (*unfold kruskal-spanning-invariant-def*, *intro conjI*)
show *symmetric g*
using *assms(1) kruskal-spanning-invariant-def* **by** *simp*
next
show $?h = ?h^T$
using *assms(1)* **by** (*simp add*: *conv-complement conv-dist-inf inf-commute inf-left-commute kruskal-spanning-invariant-def*)
next
show $g \sqcap --?h = ?h$
using *1 2* **by** (*metis* (*opaque-lifting*) *assms(1) kruskal-spanning-invariant-def inf-assoc pp-dist-inf*)
next
show *spanning-forest ?f* $(-?h \sqcap g)$
proof (*unfold spanning-forest-def*, *intro conjI*)
show *injective ?f*

```
          using 5 by simp
        next
          show acyclic ?f
            apply (rule kruskal-acyclic-inv)
            using assms(1) kruskal-spanning-invariant-def spanning-forest-def
apply simp
            apply (simp add: covector-mult-closed)
            using 8 assms(1) kruskal-spanning-invariant-def spanning-forest-def
kruskal-acyclic-inv-1 apply simp
            using 8 apply (metis comp-associative mult-left-sub-dist-sup-left
star.circ-loop-fixpoint sup-commute le-bot)
            using 6 by (simp add: p-antitone-iff)
        next
          show ?f ≤ −−(− ?h ⊓ g)
            apply (rule kruskal-subgraph-inv)
            using assms(1) kruskal-spanning-invariant-def spanning-forest-def
apply simp
            using assms(1) apply (metis kruskal-spanning-invariant-def
minarc-below order.trans pp-isotone-inf)
            using assms(1) kruskal-spanning-invariant-def apply simp
            using assms(1) kruskal-spanning-invariant-def by simp
        next
          show components (− ?h ⊓ g) ≤ forest-components ?f
            apply (rule kruskal-spanning-inv)
            using 5 apply simp
            using 1 regular-closed-star regular-conv-closed regular-mult-closed
apply simp
            using 1 apply simp
            using assms(1) kruskal-spanning-invariant-def spanning-forest-def by
simp
        next
          show regular ?f
            using 2 by simp
        qed
      next
        show ?n2 < ?n1
          using 4 by simp
      qed
    qed
  next
    show ¬ ?e ≤ − ?F ⟶ kruskal-spanning-invariant f g ?h ∧ ?n2 < ?n1
    proof
      assume ¬ ?e ≤ − ?F
      hence 9: ?e ≤ ?F
        using 2 assms(2) arc-in-partition minarc-arc by fastforce
      show kruskal-spanning-invariant f g ?h ∧ ?n2 < ?n1
      proof (unfold kruskal-spanning-invariant-def, intro conjI)
        show symmetric g
          using assms(1) kruskal-spanning-invariant-def by simp
```

```
    next
      show ?h = ?h^T
        using assms(1) by (simp add: conv-complement conv-dist-inf
inf-commute inf-left-commute kruskal-spanning-invariant-def)
    next
      show g ⊓ −−?h = ?h
        using 1 2 by (metis (opaque-lifting) assms(1)
kruskal-spanning-invariant-def inf-assoc pp-dist-inf)
    next
      show spanning-forest f (−?h ⊓ g)
      proof (unfold spanning-forest-def, intro conjI)
        show injective f
          using assms(1) kruskal-spanning-invariant-def spanning-forest-def by
simp
      next
        show acyclic f
          using assms(1) kruskal-spanning-invariant-def spanning-forest-def by
simp
      next
        have f ≤ −−(−h ⊓ g)
          using assms(1) kruskal-spanning-invariant-def spanning-forest-def by
simp
        also have ... ≤ −−(−?h ⊓ g)
          using comp-inf.mult-right-isotone inf.sup-monoid.add-commute
inf-left-commute p-antitone-inf pp-isotone by presburger
        finally show f ≤ −−(−?h ⊓ g)
          by simp
      next
        show components (−?h ⊓ g) ≤ ?F
          apply (rule kruskal-spanning-inv-1)
          using 9 apply simp
          using 1 apply simp
          using assms(1) kruskal-spanning-invariant-def spanning-forest-def
apply simp
          using assms(1) kruskal-spanning-invariant-def
forest-components-equivalence spanning-forest-def by simp
      next
        show regular f
          using 1 by simp
      qed
    next
      show ?n2 < ?n1
        using 4 by simp
    qed
  qed
 qed
qed
```

The following result shows that Kruskal's algorithm terminates and constructs a spanning tree. We cannot yet show that this is a minimum spanning

tree.

theorem *kruskal-spanning*:
VARS e f h
[*symmetric g*]
$f := bot$;
$h := g$;
WHILE $h \neq bot$
INV { *kruskal-spanning-invariant f g h* }
VAR { *card* { x . *regular* $x \wedge x \leq --h$ } }
DO $e :=$ *minarc h*;
IF $e \leq -$*forest-components f THEN*
$f := (f \sqcap -(top * e * f^{T\star})) \sqcup (f \sqcap top * e * f^{T\star})^T \sqcup e$
ELSE
SKIP
FI;
$h := h \sqcap -e \sqcap -e^T$
OD
[*spanning-forest f g*]
apply *vcg-tc-simp*
using *kruskal-vc-1* **apply** *simp*
using *kruskal-vc-2* **apply** *simp*
using *kruskal-spanning-invariant-def* **by** *auto*

Because we have shown total correctness, we conclude that a spanning tree exists.

lemma *kruskal-exists-spanning*:
symmetric $g \Longrightarrow \exists f$. *spanning-forest f g*
using *tc-extract-function kruskal-spanning* **by** *blast*

This implies that a minimum spanning tree exists, which is used in the subsequent correctness proof.

lemma *kruskal-exists-minimal-spanning*:
assumes *symmetric g*
shows $\exists f$. *minimum-spanning-forest f g*
proof −
let *?s* = { f . *spanning-forest f g* }
have $\exists m{\in}$*?s* . $\forall z{\in}$*?s* . *sum* $(m \sqcap g) \leq$ *sum* $(z \sqcap g)$
apply (*rule finite-set-minimal*)
using *finite-regular spanning-forest-def* **apply** *simp*
using *assms kruskal-exists-spanning* **apply** *simp*
using *sum-linear* **by** *simp*
thus *?thesis*
using *minimum-spanning-forest-def* **by** *simp*
qed

Kruskal's minimum spanning tree algorithm terminates and is correct. This is the same algorithm that is used in the previous correctness proof, with the same precondition and variant, but with a different invariant and postcondition.

theorem *kruskal*:
VARS e f h
[*symmetric g*]
$f := bot$;
$h := g$;
WHILE $h \neq bot$
INV { *kruskal-invariant f g h* }
VAR { *card* { x . *regular* $x \wedge x \leq --h$ } }
DO $e :=$ *minarc* h;
IF $e \leq$ *−forest-components f THEN*
$f := (f \sqcap -(top * e * f^{T\star})) \sqcup (f \sqcap top * e * f^{T\star})^T \sqcup e$
ELSE
SKIP
FI;
$h := h \sqcap -e \sqcap -e^T$
OD
[*minimum-spanning-forest f g*]
proof *vcg-tc-simp*
assume *symmetric g*
thus *kruskal-invariant bot g g*
using *kruskal-vc-1 kruskal-exists-minimal-spanning kruskal-invariant-def* **by** *simp*
next
fix *f h*
let *?e* = *minarc h*
let *?f* = $(f \sqcap -(top * ?e * f^{T\star})) \sqcup (f \sqcap top * ?e * f^{T\star})^T \sqcup ?e$
let *?h* = $h \sqcap -?e \sqcap -?e^T$
let *?F* = *forest-components f*
let *?n1* = *card* { x . *regular* $x \wedge x \leq --h$ }
let *?n2* = *card* { x . *regular* $x \wedge x \leq --h \wedge x \leq -?e \wedge x \leq -?e^T$ }
assume *1*: *kruskal-invariant f g h* $\wedge$ $h \neq bot$
from *1* **obtain** *w* **where** *2*: *minimum-spanning-forest w g* $\wedge$ $f \leq w \sqcup w^T$
using *kruskal-invariant-def* **by** *auto*
hence *3*: *regular f* $\wedge$ *regular w* $\wedge$ *regular ?e*
using *1* **by** (*metis kruskal-invariant-def kruskal-spanning-invariant-def minimum-spanning-forest-def spanning-forest-def minarc-regular*)
show (*?e* $\leq -?F \longrightarrow$ *kruskal-invariant ?f g ?h* $\wedge$ *?n2* $<$ *?n1*) $\wedge$ ($\neg$ *?e* $\leq -?F \longrightarrow$ *kruskal-invariant f g ?h* $\wedge$ *?n2* $<$ *?n1*)
proof (*rule conjI*)
show *?e* $\leq -?F \longrightarrow$ *kruskal-invariant ?f g ?h* $\wedge$ *?n2* $<$ *?n1*
proof
assume *4*: *?e* $\leq -?F$
have *5*: *equivalence ?F*
using *1 kruskal-invariant-def kruskal-spanning-invariant-def forest-components-equivalence spanning-forest-def* **by** *simp*
have $?e^T * top * ?e^T = ?e^T$
using *1* **by** (*simp add*: *arc-top-arc minarc-arc*)
hence $?e^T * top * ?e^T \leq -?F$
using *4 5 conv-complement conv-isotone* **by** *fastforce*

```
    hence 6: ?e * ?F * ?e = bot
      using le-bot triple-schroeder-p by simp
    show kruskal-invariant ?f g ?h ∧ ?n2 < ?n1
    proof (unfold kruskal-invariant-def, intro conjI)
      show kruskal-spanning-invariant ?f g ?h
        using 1 4 kruskal-vc-2 kruskal-invariant-def by simp
    next
      show ∃ w . minimum-spanning-forest w g ∧ ?f ≤ w ⊔ w^T
      proof
        let ?p = w ⊓ top * ?e * w^T⋆
        let ?v = (w ⊓ -(top * ?e * w^T⋆)) ⊔ ?p^T
        have 7: regular ?p
          using 3 regular-closed-star regular-conv-closed regular-mult-closed by
simp
        have 8: injective ?v
          apply (rule kruskal-exchange-injective-inv-1)
          using 2 minimum-spanning-forest-def spanning-forest-def apply simp
          apply (simp add: covector-mult-closed)
          apply (simp add: comp-associative comp-isotone
star.right-plus-below-circ)
          using 1 2 kruskal-injective-inv-3 minarc-arc
minimum-spanning-forest-def spanning-forest-def by simp
        have 9: components g ≤ forest-components ?v
          apply (rule kruskal-exchange-spanning-inv-1)
          using 8 apply simp
          using 7 apply simp
          using 2 minimum-spanning-forest-def spanning-forest-def by simp
        have 10: spanning-forest ?v g
        proof (unfold spanning-forest-def, intro conjI)
          show injective ?v
            using 8 by simp
        next
          show acyclic ?v
            apply (rule kruskal-exchange-acyclic-inv-1)
            using 2 minimum-spanning-forest-def spanning-forest-def apply simp
            by (simp add: covector-mult-closed)
        next
          show ?v ≤ --g
            apply (rule sup-least)
            using 2 inf.coboundedI1 minimum-spanning-forest-def
spanning-forest-def apply simp
            using 1 2 by (metis kruskal-invariant-def
kruskal-spanning-invariant-def conv-complement conv-dist-inf order.trans
inf.absorb2 inf.cobounded1 minimum-spanning-forest-def spanning-forest-def)
        next
          show components g ≤ forest-components ?v
            using 9 by simp
        next
          show regular ?v
```

using *3 regular-closed-star regular-conv-closed regular-mult-closed* **by** *simp*

qed

have *11*: $sum\ (?v \sqcap g) = sum\ (w \sqcap g)$

proof −

have $sum\ (?v \sqcap g) = sum\ (w \sqcap -(top * ?e * w^{T\star}) \sqcap g) + sum\ (?p^T \sqcap g)$

using *2* **by** (*metis conv-complement conv-top epm-8 inf-import-p inf-top-right regular-closed-top vector-top-closed minimum-spanning-forest-def spanning-forest-def sum-disjoint*)

also have ... $= sum\ (w \sqcap -(top * ?e * w^{T\star}) \sqcap g) + sum\ (?p \sqcap g)$

using *1 kruskal-invariant-def kruskal-spanning-invariant-def sum-symmetric* **by** *simp*

also have ... $= sum\ (((w \sqcap -(top * ?e * w^{T\star})) \sqcup ?p) \sqcap g)$

using *inf-commute inf-left-commute sum-disjoint* **by** *simp*

also have ... $= sum\ (w \sqcap g)$

using *3 7 maddux-3-11-pp* **by** *simp*

finally show *?thesis*

by *simp*

qed

have *12*: $?v \sqcup ?v^T = w \sqcup w^T$

proof −

have $?v \sqcup ?v^T = (w \sqcap -?p) \sqcup ?p^T \sqcup (w^T \sqcap -?p^T) \sqcup ?p$

using *conv-complement conv-dist-inf conv-dist-sup inf-import-p sup-assoc* **by** *simp*

also have ... $= w \sqcup w^T$

using *3 7 conv-complement conv-dist-inf inf-import-p maddux-3-11-pp sup-monoid.add-assoc sup-monoid.add-commute* **by** *simp*

finally show *?thesis*

by *simp*

qed

have *13*: $?v * ?e^T = bot$

apply (*rule kruskal-reroot-edge*)

using *1* **apply** (*simp add*: *minarc-arc*)

using *2 minimum-spanning-forest-def spanning-forest-def* **by** *simp*

have $?v \sqcap ?e \leq ?v \sqcap top * ?e$

using *inf.sup-right-isotone top-left-mult-increasing* **by** *simp*

also have ... $\leq ?v * (top * ?e)^T$

using *covector-restrict-comp-conv covector-mult-closed vector-top-closed* **by** *simp*

finally have *14*: $?v \sqcap ?e = bot$

using *13* **by** (*metis conv-dist-comp mult-assoc le-bot mult-left-zero*)

let $?d = ?v \sqcap top * ?e^T * ?v^{T\star} \sqcap ?F * ?e^T * top \sqcap top * ?e * -?F$

let $?w = (?v \sqcap -?d) \sqcup ?e$

have *15*: *regular ?d*

using *3 regular-closed-star regular-conv-closed regular-mult-closed* **by** *simp*

have *16*: $?F \leq -?d$

apply (*rule kruskal-edge-between-components-1*)

using *5* **apply** *simp*
using *1 conv-dist-comp minarc-arc mult-assoc* **by** *simp*
have *17*: $f \sqcup f^T \leq (?v \sqcap -?d \sqcap -?d^T) \sqcup (?v^T \sqcap -?d \sqcap -?d^T)$
apply (*rule kruskal-edge-between-components-2*)
using *16* **apply** *simp*
using *1 kruskal-invariant-def kruskal-spanning-invariant-def spanning-forest-def* **apply** *simp*
using *2 12* **by** (*metis conv-dist-sup conv-involutive conv-isotone le-supI sup-commute*)
show *minimum-spanning-forest ?w g* $\wedge$ $?f \leq ?w \sqcup ?w^T$
proof (*intro conjI*)
have *18*: $?e^T \leq ?v^\star$
apply (*rule kruskal-edge-arc-1*[**where** *g*=*g* **and** *h*=*h*])
using *minarc-below* **apply** *simp*
using *1* **apply** (*metis kruskal-invariant-def kruskal-spanning-invariant-def inf-le1*)
using *1 kruskal-invariant-def kruskal-spanning-invariant-def* **apply** *simp*
using *9* **apply** *simp*
using *13* **by** *simp*
have *19*: *arc ?d*
apply (*rule kruskal-edge-arc*)
using *5* **apply** *simp*
using *10 spanning-forest-def* **apply** *blast*
using *1* **apply** (*simp add*: *minarc-arc*)
using *3* **apply** (*metis conv-complement pp-dist-star regular-mult-closed*)
using *2 8 12* **apply** (*simp add*: *kruskal-forest-components-inf*)
using *10 spanning-forest-def* **apply** *simp*
using *13* **apply** *simp*
using *6* **apply** *simp*
using *18* **by** *simp*
show *minimum-spanning-forest ?w g*
proof (*unfold minimum-spanning-forest-def*, *intro conjI*)
have $(?v \sqcap -?d) * ?e^T \leq ?v * ?e^T$
using *inf-le1 mult-left-isotone* **by** *simp*
hence $(?v \sqcap -?d) * ?e^T = bot$
using *13 le-bot* **by** *simp*
hence *20*: $?e * (?v \sqcap -?d)^T = bot$
using *conv-dist-comp conv-involutive conv-bot* **by** *force*
have *21*: *injective ?w*
apply (*rule injective-sup*)
using *8* **apply** (*simp add*: *injective-inf-closed*)
using *20* **apply** *simp*
using *1 arc-injective minarc-arc* **by** *blast*
show *spanning-forest ?w g*
proof (*unfold spanning-forest-def*, *intro conjI*)
show *injective ?w*
using *21* **by** *simp*

```
      next
        show acyclic ?w
          apply (rule kruskal-exchange-acyclic-inv-2)
          using 10 spanning-forest-def apply blast
          using 8 apply simp
          using inf.coboundedI1 apply simp
          using 19 apply simp
          using 1 apply (simp add: minarc-arc)
          using inf.cobounded2 inf.coboundedI1 apply simp
          using 13 by simp
        next
        have ?w ≤ ?v ⊔ ?e
          using inf-le1 sup-left-isotone by simp
        also have ... ≤ −−g ⊔ ?e
          using 10 sup-left-isotone spanning-forest-def by blast
        also have ... ≤ −−g ⊔ −−h
          by (simp add: le-supI2 minarc-below)
        also have ... = −−g
          using 1 by (metis kruskal-invariant-def
kruskal-spanning-invariant-def pp-isotone-inf sup.orderE)
        finally show ?w ≤ −−g
          by simp
      next
        have 22: ?d ≤ (?v ⊓ −?d)^T* ∗ ?e^T ∗ top
          apply (rule kruskal-exchange-spanning-inv-2)
          using 8 apply simp
          using 13 apply (metis semiring.mult-not-zero star-absorb
star-simulation-right-equal)
          using 17 apply simp
          by (simp add: inf.coboundedI1)
        have components g ≤ forest-components ?v
          using 10 spanning-forest-def by auto
        also have ... ≤ forest-components ?w
          apply (rule kruskal-exchange-forest-components-inv)
          using 21 apply simp
          using 15 apply simp
          using 1 apply (simp add: arc-top-arc minarc-arc)
          apply (simp add: inf.coboundedI1)
          using 13 apply simp
          using 8 apply simp
          apply (simp add: le-infI1)
          using 22 by simp
        finally show components g ≤ forest-components ?w
          by simp
      next
        show regular ?w
          using 3 7 regular-conv-closed by simp
      qed
    next
```

have *23*: *?e* ⊓ *g* ≠ *bot*
using *1* **by** (*metis kruskal-invariant-def kruskal-spanning-invariant-def comp-inf.semiring.mult-zero-right inf.sup-monoid.add-assoc inf.sup-monoid.add-commute minarc-bot-iff minarc-meet-bot*)
have *g* ⊓ −*h* ≤ (*g* ⊓ −*h*)⋆
using *star.circ-increasing* **by** *simp*
also have ... ≤ (−−(*g* ⊓ −*h*))⋆
using *pp-increasing star-isotone* **by** *blast*
also have ... ≤ *?F*
using *1 kruskal-invariant-def kruskal-spanning-invariant-def inf.sup-monoid.add-commute spanning-forest-def* **by** *simp*
finally have *24*: *g* ⊓ −*h* ≤ *?F*
by *simp*
have *?d* ≤ −−*g*
using *10 inf.coboundedI1 spanning-forest-def* **by** *blast*
hence *?d* ≤ −−*g* ⊓ −*?F*
using *16 inf.boundedI p-antitone-iff* **by** *simp*
also have ... = −−(*g* ⊓ −*?F*)
by *simp*
also have ... ≤ −−*h*
using *24 p-shunting-swap pp-isotone* **by** *fastforce*
finally have *25*: *?d* ≤ −−*h*
by *simp*
have *?d* = *bot* ⟶ *top* = *bot*
using *19* **by** (*metis mult-left-zero mult-right-zero*)
hence *?d* ≠ *bot*
using *1 le-bot* **by** *auto*
hence *26*: *?d* ⊓ *h* ≠ *bot*
using *25* **by** (*metis inf.absorb-iff2 inf-commute pseudo-complement*)
have *sum* (*?e* ⊓ *g*) = *sum* (*?e* ⊓ −−*h* ⊓ *g*)
by (*simp add*: *inf.absorb1 minarc-below*)
also have ... = *sum* (*?e* ⊓ *h*)
using *1* **by** (*metis kruskal-invariant-def kruskal-spanning-invariant-def inf.left-commute inf.sup-monoid.add-commute*)
also have ... ≤ *sum* (*?d* ⊓ *h*)
using *19 26 minarc-min* **by** *simp*
also have ... = *sum* (*?d* ⊓ (−−*h* ⊓ *g*))
using *1 kruskal-invariant-def kruskal-spanning-invariant-def inf-commute* **by** *simp*
also have ... = *sum* (*?d* ⊓ *g*)
using *25* **by** (*simp add*: *inf.absorb2 inf-assoc inf-commute*)
finally have *27*: *sum* (*?e* ⊓ *g*) ≤ *sum* (*?d* ⊓ *g*)
by *simp*
have *?v* ⊓ *?e* ⊓ −*?d* = *bot*
using *14* **by** *simp*
hence *sum* (*?w* ⊓ *g*) = *sum* (*?v* ⊓ −*?d* ⊓ *g*) + *sum* (*?e* ⊓ *g*)
using *sum-disjoint inf-commute inf-assoc* **by** *simp*
also have ... ≤ *sum* (*?v* ⊓ −*?d* ⊓ *g*) + *sum* (*?d* ⊓ *g*)

```
            using 23 27 sum-plus-right-isotone by simp
          also have ... = sum (((?v ⊓ −?d) ⊔ ?d) ⊓ g)
            using sum-disjoint inf-le2 pseudo-complement by simp
          also have ... = sum ((?v ⊔ ?d) ⊓ (−?d ⊔ ?d) ⊓ g)
            by (simp add: sup-inf-distrib2)
          also have ... = sum ((?v ⊔ ?d) ⊓ g)
            using 15 by (metis inf-top-right stone)
          also have ... = sum (?v ⊓ g)
            by (simp add: inf.sup-monoid.add-assoc)
          finally have sum (?w ⊓ g) ≤ sum (?v ⊓ g)
            by simp
          thus ∀ u . spanning-forest u g ⟶ sum (?w ⊓ g) ≤ sum (u ⊓ g)
            using 2 11 minimum-spanning-forest-def by auto
        qed
      next
        have ?f ≤ f ⊔ f^T ⊔ ?e
          using conv-dist-inf inf-le1 sup-left-isotone sup-mono by presburger
        also have ... ≤ (?v ⊓ −?d ⊓ −?d^T) ⊔ (?v^T ⊓ −?d ⊓ −?d^T) ⊔ ?e
          using 17 sup-left-isotone by simp
        also have ... ≤ (?v ⊓ −?d) ⊔ (?v^T ⊓ −?d ⊓ −?d^T) ⊔ ?e
          using inf.cobounded1 sup-inf-distrib2 by presburger
        also have ... = ?w ⊔ (?v^T ⊓ −?d ⊓ −?d^T)
          by (simp add: sup-assoc sup-commute)
        also have ... ≤ ?w ⊔ (?v^T ⊓ −?d^T)
          using inf.sup-right-isotone inf-assoc sup-right-isotone by simp
        also have ... ≤ ?w ⊔ ?w^T
          using conv-complement conv-dist-inf conv-dist-sup sup-right-isotone
by simp
        finally show ?f ≤ ?w ⊔ ?w^T
          by simp
      qed
    qed
  next
    show ?n2 < ?n1
      using 1 kruskal-vc-2 kruskal-invariant-def by auto
  qed
 qed
next
 show ¬ ?e ≤ −?F ⟶ kruskal-invariant f g ?h ∧ ?n2 < ?n1
  using 1 kruskal-vc-2 kruskal-invariant-def by auto
qed
next
 fix f
 assume 28: kruskal-invariant f g bot
 hence 29: spanning-forest f g
  using kruskal-invariant-def kruskal-spanning-invariant-def by auto
 from 28 obtain w where 30: minimum-spanning-forest w g ∧ f ≤ w ⊔ w^T
  using kruskal-invariant-def by auto
 hence w = w ⊓ −−g
```

by (*simp add*: *inf.absorb1 minimum-spanning-forest-def spanning-forest-def*)
also have ... $\leq w \sqcap$ *components g*
by (*metis inf.sup-right-isotone star.circ-increasing*)
also have ... $\leq w \sqcap f^{T\star} * f^\star$
using *29 spanning-forest-def inf.sup-right-isotone* **by** *simp*
also have ... $\leq f \sqcup f^T$
apply (*rule cancel-separate-6*[**where** $z{=}w$ **and** $y{=}w^T$])
using *30 minimum-spanning-forest-def spanning-forest-def* **apply** *simp*
using *30* **apply** (*metis conv-dist-inf conv-dist-sup conv-involutive inf.cobounded2 inf.orderE*)
using *30* **apply** (*simp add*: *sup-commute*)
using *30 minimum-spanning-forest-def spanning-forest-def* **apply** *simp*
using *30* **by** (*metis acyclic-star-below-complement comp-inf.mult-right-isotone inf-p le-bot minimum-spanning-forest-def spanning-forest-def*)
finally have *31*: $w \leq f \sqcup f^T$
by *simp*
have *sum* $(f \sqcap g)$ = *sum* $((w \sqcup w^T) \sqcap (f \sqcap g))$
using *30* **by** (*metis inf-absorb2 inf.assoc*)
also have ... = *sum* $(w \sqcap (f \sqcap g))$ + *sum* $(w^T \sqcap (f \sqcap g))$
using *30 inf.commute acyclic-asymmetric sum-disjoint minimum-spanning-forest-def spanning-forest-def* **by** *simp*
also have ... = *sum* $(w \sqcap (f \sqcap g))$ + *sum* $(w \sqcap (f^T \sqcap g^T))$
by (*metis conv-dist-inf conv-involutive sum-conv*)
also have ... = *sum* $(f \sqcap (w \sqcap g))$ + *sum* $(f^T \sqcap (w \sqcap g))$
using *28 inf.commute inf.assoc kruskal-invariant-def kruskal-spanning-invariant-def* **by** *simp*
also have ... = *sum* $((f \sqcup f^T) \sqcap (w \sqcap g))$
using *29 acyclic-asymmetric inf.sup-monoid.add-commute sum-disjoint spanning-forest-def* **by** *simp*
also have ... = *sum* $(w \sqcap g)$
using *31* **by** (*metis inf-absorb2 inf.assoc*)
finally show *minimum-spanning-forest f g*
using *29 30 minimum-spanning-forest-def* **by** *simp*
qed

end

end

3 Prim's Minimum Spanning Tree Algorithm

In this theory we prove total correctness of Prim's minimum spanning tree algorithm. The proof has the same overall structure as the total-correctness proof of Kruskal's algorithm [6]. The partial-correctness proof of Prim's algorithm is discussed in [3, 5].

theory *Prim*

imports *HOL−Hoare.Hoare-Logic Aggregation-Algebras.Aggregation-Algebras*

begin

context *m-kleene-algebra*
begin

abbreviation *component g r* ≡ $r^T * (--g)^\star$
definition *spanning-tree t g r* ≡ *forest t* ∧ *t* ≤ (*component g r*)^T ∗ (*component g r*) ⊓ −−*g* ∧ *component g r* ≤ $r^T * t^\star$ ∧ *regular t*
definition *minimum-spanning-tree t g r* ≡ *spanning-tree t g r* ∧ (∀ *u* . *spanning-tree u g r* ⟶ *sum* (*t* ⊓ *g*) ≤ *sum* (*u* ⊓ *g*))
definition *prim-precondition g r* ≡ *g* = g^T ∧ *injective r* ∧ *vector r* ∧ *regular r*
definition *prim-spanning-invariant t v g r* ≡ *prim-precondition g r* ∧ $v^T = r^T * t^\star$ ∧ *spanning-tree t* (*v* ∗ v^T ⊓ *g*) *r*
definition *prim-invariant t v g r* ≡ *prim-spanning-invariant t v g r* ∧ (∃ *w* . *minimum-spanning-tree w g r* ∧ *t* ≤ *w*)

lemma *span-tree-split*:
assumes *vector r*
shows *t* ≤ (*component g r*)^T ∗ (*component g r*) ⊓ −−*g* ⟷ (*t* ≤ (*component g r*)^T ∧ *t* ≤ *component g r* ∧ *t* ≤ −−*g*)
proof −
have (*component g r*)^T ∗ (*component g r*) = (*component g r*)^T ⊓ *component g r*
by (*metis assms conv-involutive covector-mult-closed vector-conv-covector vector-covector*)
thus *?thesis*
by *simp*
qed

lemma *span-tree-component*:
assumes *spanning-tree t g r*
shows *component g r* = *component t r*
using *assms* **by** (*simp add*: *order.antisym mult-right-isotone star-isotone spanning-tree-def*)

We first show three verification conditions which are used in both correctness proofs.

lemma *prim-vc-1*:
assumes *prim-precondition g r*
shows *prim-spanning-invariant bot r g r*
proof (*unfold prim-spanning-invariant-def*, *intro conjI*)
show *prim-precondition g r*
using *assms* **by** *simp*
next
show $r^T = r^T * bot^\star$
by (*simp add*: *star-absorb*)
next
let *?ss* = *r* ∗ r^T ⊓ *g*
show *spanning-tree bot ?ss r*

proof (*unfold spanning-tree-def*, *intro conjI*)
show *injective bot*
by *simp*
next
show *acyclic bot*
by *simp*
next
show $bot \leq (component\ ?ss\ r)^T * (component\ ?ss\ r) \sqcap --?ss$
by *simp*
next
have $component\ ?ss\ r \leq component\ (r * r^T)\ r$
by (*simp add*: *mult-right-isotone star-isotone*)
also have $... \leq r^T * 1^\star$
using *assms* **by** (*metis order.eq-iff p-antitone regular-one-closed star-sub-one prim-precondition-def*)
also have $... = r^T * bot^\star$
by (*simp add*: *star.circ-zero star-one*)
finally show $component\ ?ss\ r \leq r^T * bot^\star$
.
next
show *regular bot*
by *simp*
qed
qed

lemma *prim-vc-2*:
assumes *prim-spanning-invariant t v g r*
and $v * -v^T \sqcap g \neq bot$
shows *prim-spanning-invariant* $(t \sqcup minarc\ (v * -v^T \sqcap g))\ (v \sqcup minarc\ (v * -v^T \sqcap g)^T * top)\ g\ r \wedge card\ \{\ x\ .\ regular\ x \wedge x \leq component\ g\ r \wedge x \leq -(v \sqcup minarc\ (v * -v^T \sqcap g)^T * top)^T\ \} < card\ \{\ x\ .\ regular\ x \wedge x \leq component\ g\ r \wedge x \leq -v^T\ \}$
proof −
let $?vcv = v * -v^T \sqcap g$
let $?e = minarc\ ?vcv$
let $?t = t \sqcup ?e$
let $?v = v \sqcup ?e^T * top$
let $?c = component\ g\ r$
let $?g = --g$
let $?n1 = card\ \{\ x\ .\ regular\ x \wedge x \leq ?c \wedge x \leq -v^T\ \}$
let $?n2 = card\ \{\ x\ .\ regular\ x \wedge x \leq ?c \wedge x \leq -?v^T\ \}$
have *1*: $regular\ v \wedge regular\ (v * v^T) \wedge regular\ (?v * ?v^T) \wedge regular\ (top * ?e)$
using *assms(1)* **by** (*metis prim-spanning-invariant-def spanning-tree-def prim-precondition-def regular-conv-closed regular-closed-star regular-mult-closed conv-involutive regular-closed-top regular-closed-sup minarc-regular*)
hence *2*: $t \leq v * v^T \sqcap ?g$
using *assms(1)* **by** (*metis prim-spanning-invariant-def spanning-tree-def inf-pp-commute inf.boundedE*)
hence *3*: $t \leq v * v^T$

by *simp*
have *4*: $t \leq ?g$
using *2* **by** *simp*
have *5*: $?e \leq v * -v^T \sqcap ?g$
using *1* **by** (*metis minarc-below pp-dist-inf regular-mult-closed regular-closed-p*)
hence *6*: $?e \leq v * -v^T$
by *simp*
have *7*: *vector v*
using *assms*(*1*) *prim-spanning-invariant-def prim-precondition-def* **by** (*simp add*: *covector-mult-closed vector-conv-covector*)
hence *8*: $?e \leq v$
using *6* **by** (*metis conv-complement inf.boundedE vector-complement-closed vector-covector*)
have *9*: $?e * t = bot$
using *3 6 7 et*(*1*) **by** *blast*
have *10*: $?e * t^T = bot$
using *3 6 7 et*(*2*) **by** *simp*
have *11*: *arc ?e*
using *assms*(*2*) *minarc-arc* **by** *simp*
have $r^T \leq r^T * t^\star$
by (*metis mult-right-isotone order-refl semiring.mult-not-zero star.circ-separate-mult-1 star-absorb*)
hence *12*: $r^T \leq v^T$
using *assms*(*1*) **by** (*simp add*: *prim-spanning-invariant-def*)
have *13*: *vector r* $\wedge$ *injective r* $\wedge$ $v^T = r^T * t^\star$
using *assms*(*1*) *prim-spanning-invariant-def prim-precondition-def minimum-spanning-tree-def spanning-tree-def reachable-restrict* **by** *simp*
have $g = g^T$
using *assms*(*1*) *prim-invariant-def prim-spanning-invariant-def prim-precondition-def* **by** *simp*
hence *14*: $?g^T = ?g$
using *conv-complement* **by** *simp*
show *prim-spanning-invariant ?t ?v g r* $\wedge$ *?n2* $<$ *?n1*
proof (*rule conjI*)
show *prim-spanning-invariant ?t ?v g r*
proof (*unfold prim-spanning-invariant-def*, *intro conjI*)
show *prim-precondition g r*
using *assms*(*1*) *prim-spanning-invariant-def* **by** *simp*
next
show $?v^T = r^T * ?t^\star$
using *assms*(*1*) *6 7 9* **by** (*simp add*: *reachable-inv prim-spanning-invariant-def prim-precondition-def spanning-tree-def*)
next
let $?G = ?v * ?v^T \sqcap g$
show *spanning-tree ?t ?G r*
proof (*unfold spanning-tree-def*, *intro conjI*)
show *injective ?t*
using *assms*(*1*) *10 11* **by** (*simp add*: *injective-inv*

prim-spanning-invariant-def spanning-tree-def)
next
show *acyclic ?t*
using *assms*(*1*) *3 6 7 acyclic-inv prim-spanning-invariant-def spanning-tree-def* **by** *simp*
next
show *?t* ≤ (*component ?G r*)^T ∗ (*component ?G r*) ⊓ −−*?G*
using *1 2 5 7 13 prim-subgraph-inv inf-pp-commute mst-subgraph-inv-2* **by** *auto*
next
show *component* (*?v* ∗ *?v*^{T} ⊓ *g*) *r* ≤ *r*^{T} ∗ *?t**
proof −
have *15*: *r*^{T} ∗ (*v* ∗ *v*^{T} ⊓ *?g*)* ≤ *r*^{T} ∗ *t**
using *assms*(*1*) *1* **by** (*metis prim-spanning-invariant-def spanning-tree-def inf-pp-commute*)
have *component* (*?v* ∗ *?v*^{T} ⊓ *g*) *r* = *r*^{T} ∗ (*?v* ∗ *?v*^{T} ⊓ *?g*)*
using *1* **by** *simp*
also have ... ≤ *r*^{T} ∗ *?t**
using *2 6 7 11 12 13 14 15* **by** (*metis span-inv*)
finally show *?thesis*
.
qed
next
show *regular ?t*
using *assms*(*1*) **by** (*metis prim-spanning-invariant-def spanning-tree-def regular-closed-sup minarc-regular*)
qed
qed
next
have *16*: *top* ∗ *?e* ≤ *?c*
proof −
have *top* ∗ *?e* = *top* ∗ *?e*^{T} ∗ *?e*
using *11* **by** (*metis arc-top-edge mult-assoc*)
also have ... ≤ *v*^{T} ∗ *?e*
using *7 8* **by** (*metis conv-dist-comp conv-isotone mult-left-isotone symmetric-top-closed*)
also have ... ≤ *v*^{T} ∗ *?g*
using *5 mult-right-isotone* **by** *auto*
also have ... = *r*^{T} ∗ *t** ∗ *?g*
using *13* **by** *simp*
also have ... ≤ *r*^{T} ∗ *?g** ∗ *?g*
using *4* **by** (*simp add*: *mult-left-isotone mult-right-isotone star-isotone*)
also have ... ≤ *?c*
by (*simp add*: *comp-associative mult-right-isotone star.right-plus-below-circ*)
finally show *?thesis*
by *simp*
qed
have *17*: *top* ∗ *?e* ≤ −*v*^{T}
using *6 7* **by** (*simp add*: *schroeder-4-p vTeT*)

```
    have 18: ¬ top ∗ ?e ≤ −(top ∗ ?e)
      by (metis assms(2) inf.orderE minarc-bot-iff conv-complement-sub-inf inf-p
inf-top.left-neutral p-bot symmetric-top-closed vector-top-closed)
    have 19: −?v^T = −v^T ⊓ −(top ∗ ?e)
      by (simp add: conv-dist-comp conv-dist-sup)
    hence 20: ¬ top ∗ ?e ≤ −?v^T
      using 18 by simp
    show ?n2 < ?n1
      apply (rule psubset-card-mono)
      using finite-regular apply simp
      using 1 16 17 19 20 by auto
  qed
qed

lemma prim-vc-3:
  assumes prim-spanning-invariant t v g r
      and v ∗ −v^T ⊓ g = bot
    shows spanning-tree t g r
proof −
  let ?g = −−g
  have 1: regular v ∧ regular (v ∗ v^T)
    using assms(1) by (metis prim-spanning-invariant-def spanning-tree-def
prim-precondition-def regular-conv-closed regular-closed-star regular-mult-closed
conv-involutive)
  have 2: v ∗ −v^T ⊓ ?g = bot
    using assms(2) pp-inf-bot-iff pp-pp-inf-bot-iff by simp
  have 3: v^T = r^T ∗ t^⋆ ∧ vector v
    using assms(1) by (simp add: covector-mult-closed prim-invariant-def
prim-spanning-invariant-def vector-conv-covector prim-precondition-def)
  have 4: t ≤ v ∗ v^T ⊓ ?g
    using assms(1) 1 by (metis prim-spanning-invariant-def inf-pp-commute
spanning-tree-def inf.boundedE)
  have r^T ∗ (v ∗ v^T ⊓ ?g)^⋆ ≤ r^T ∗ t^⋆
    using assms(1) 1 by (metis prim-spanning-invariant-def inf-pp-commute
spanning-tree-def)
  hence 5: component g r = v^T
    using 1 2 3 4 by (metis span-post)
  have regular (v ∗ v^T)
    using assms(1) by (metis prim-spanning-invariant-def spanning-tree-def
prim-precondition-def regular-conv-closed regular-closed-star regular-mult-closed
conv-involutive)
  hence 6: t ≤ v ∗ v^T ⊓ ?g
    by (metis assms(1) prim-spanning-invariant-def spanning-tree-def
inf-pp-commute inf.boundedE)
  show spanning-tree t g r
    apply (unfold spanning-tree-def, intro conjI)
    using assms(1) prim-spanning-invariant-def spanning-tree-def apply simp
    using assms(1) prim-spanning-invariant-def spanning-tree-def apply simp
    using 5 6 apply simp
```

using *assms*(*1*) *5 prim-spanning-invariant-def* **apply** *simp*
using *assms*(*1*) *prim-spanning-invariant-def spanning-tree-def* **by** *simp*
qed

The following result shows that Prim's algorithm terminates and constructs a spanning tree. We cannot yet show that this is a minimum spanning tree.

theorem *prim-spanning*:
VARS t v e
[*prim-precondition g r*]
t := *bot*;
v := *r*;
WHILE $v * -v^T \sqcap g \neq bot$
INV { *prim-spanning-invariant t v g r* }
VAR { *card* { *x* . *regular x* ∧ *x* ≤ *component g r* ⊓ $-v^T$ } }
DO e := *minarc* ($v * -v^T \sqcap g$);
t := $t \sqcup e$;
v := $v \sqcup e^T * top$
OD
[*spanning-tree t g r*]
apply *vcg-tc-simp*
apply (*simp add*: *prim-vc-1*)
using *prim-vc-2* **apply** *blast*
using *prim-vc-3* **by** *auto*

Because we have shown total correctness, we conclude that a spanning tree exists.

lemma *prim-exists-spanning*:
prim-precondition g r ⟹ ∃ *t* . *spanning-tree t g r*
using *tc-extract-function prim-spanning* **by** *blast*

This implies that a minimum spanning tree exists, which is used in the subsequent correctness proof.

lemma *prim-exists-minimal-spanning*:
assumes *prim-precondition g r*
shows ∃ *t* . *minimum-spanning-tree t g r*
proof −
let *?s* = { *t* . *spanning-tree t g r* }
have ∃ *m*∈*?s* . ∀ *z*∈*?s* . *sum* (*m* ⊓ *g*) ≤ *sum* (*z* ⊓ *g*)
apply (*rule finite-set-minimal*)
using *finite-regular spanning-tree-def* **apply** *simp*
using *assms prim-exists-spanning* **apply** *simp*
using *sum-linear* **by** *simp*
thus *?thesis*
using *minimum-spanning-tree-def* **by** *simp*
qed

Prim's minimum spanning tree algorithm terminates and is correct. This is the same algorithm that is used in the previous correctness proof, with

the same precondition and variant, but with a different invariant and post-condition.

theorem *prim*:
VARS t v e
[*prim-precondition g r* ∧ (∃ *w* . *minimum-spanning-tree w g r*)]
t := *bot*;
v := *r*;
WHILE v ∗ $-v^T$ ⊓ *g* ≠ *bot*
INV { *prim-invariant t v g r* }
VAR { *card* { *x* . *regular x* ∧ *x* ≤ *component g r* ⊓ $-v^T$ } }
DO e := *minarc* (*v* ∗ $-v^T$ ⊓ *g*);
t := *t* ⊔ *e*;
v := *v* ⊔ e^T ∗ *top*
OD
[*minimum-spanning-tree t g r*]
proof *vcg-tc-simp*
assume *prim-precondition g r* ∧ (∃ *w* . *minimum-spanning-tree w g r*)
thus *prim-invariant bot r g r*
using *prim-invariant-def prim-vc-1* **by** *simp*
next
fix *t v*
let *?vcv* = *v* ∗ $-v^T$ ⊓ *g*
let *?vv* = *v* ∗ v^T ⊓ *g*
let *?e* = *minarc ?vcv*
let *?t* = *t* ⊔ *?e*
let *?v* = *v* ⊔ $?e^T$ ∗ *top*
let *?c* = *component g r*
let *?g* = −−*g*
let *?n1* = *card* { *x* . *regular x* ∧ *x* ≤ *?c* ∧ *x* ≤ $-v^T$ }
let *?n2* = *card* { *x* . *regular x* ∧ *x* ≤ *?c* ∧ *x* ≤ $-?v^T$ }
assume *1*: *prim-invariant t v g r* ∧ *?vcv* ≠ *bot*
hence *2*: *regular v* ∧ *regular* (*v* ∗ v^T)
by (*metis* (*no-types*, *opaque-lifting*) *prim-invariant-def prim-spanning-invariant-def spanning-tree-def prim-precondition-def regular-conv-closed regular-closed-star regular-mult-closed conv-involutive*)
have *3*: *t* ≤ *v* ∗ v^T ⊓ *?g*
using *1 2* **by** (*metis* (*no-types*, *opaque-lifting*) *prim-invariant-def prim-spanning-invariant-def spanning-tree-def inf-pp-commute inf.boundedE*)
hence *4*: *t* ≤ *v* ∗ v^T
by *simp*
have *5*: *t* ≤ *?g*
using *3* **by** *simp*
have *6*: *?e* ≤ *v* ∗ $-v^T$ ⊓ *?g*
using *2* **by** (*metis minarc-below pp-dist-inf regular-mult-closed regular-closed-p*)
hence *7*: *?e* ≤ *v* ∗ $-v^T$
by *simp*
have *8*: *vector v*
using *1 prim-invariant-def prim-spanning-invariant-def prim-precondition-def*

by (*simp add*: *covector-mult-closed vector-conv-covector*)
have *9*: *arc ?e*
using *1 minarc-arc* **by** *simp*
from *1* **obtain** *w* **where** *10*: *minimum-spanning-tree w g r* $\wedge$ *t* $\leq$ *w*
by (*metis prim-invariant-def*)
hence *11*: *vector r* $\wedge$ *injective r* $\wedge$ $v^T = r^T * t^\star$ $\wedge$ *forest w* $\wedge$ *t* $\leq$ *w* $\wedge$ *w* $\leq$ *?c*^{T} $*$ *?c* $\sqcap$ *?g* $\wedge$ $r^T * (?c^T * ?c \sqcap ?g)^\star \leq r^T * w^\star$
using *1 2 prim-invariant-def prim-spanning-invariant-def prim-precondition-def minimum-spanning-tree-def spanning-tree-def reachable-restrict* **by** *simp*
hence *12*: $w * v \leq v$
using *predecessors-reachable reachable-restrict* **by** *auto*
have *13*: $g = g^T$
using *1 prim-invariant-def prim-spanning-invariant-def prim-precondition-def* **by** *simp*
hence *14*: *?g*^{T} = *?g*
using *conv-complement* **by** *simp*
show *prim-invariant ?t ?v g r* $\wedge$ *?n2* < *?n1*
proof (*unfold prim-invariant-def*, *intro conjI*)
show *prim-spanning-invariant ?t ?v g r*
using *1 prim-invariant-def prim-vc-2* **by** *blast*
next
show $\exists\, w$. *minimum-spanning-tree w g r* $\wedge$ *?t* $\leq$ *w*
proof
let *?f* = $w \sqcap v * -v^T \sqcap top * ?e * w^{T\star}$
let *?p* = $w \sqcap -v * -v^T \sqcap top * ?e * w^{T\star}$
let *?fp* = $w \sqcap -v^T \sqcap top * ?e * w^{T\star}$
let *?w* = (*w* $\sqcap$ $-$*?fp*) $\sqcup$ *?p*^{T} $\sqcup$ *?e*
have *15*: *regular ?f* $\wedge$ *regular ?fp* $\wedge$ *regular ?w*
using *2 10* **by** (*metis regular-conv-closed regular-closed-star regular-mult-closed regular-closed-top regular-closed-inf regular-closed-sup minarc-regular minimum-spanning-tree-def spanning-tree-def*)
show *minimum-spanning-tree ?w g r* $\wedge$ *?t* $\leq$ *?w*
proof (*intro conjI*)
show *minimum-spanning-tree ?w g r*
proof (*unfold minimum-spanning-tree-def*, *intro conjI*)
show *spanning-tree ?w g r*
proof (*unfold spanning-tree-def*, *intro conjI*)
show *injective ?w*
using *7 8 9 11 exchange-injective* **by** *blast*
next
show *acyclic ?w*
using *7 8 11 12 exchange-acyclic* **by** *blast*
next
show *?w* $\leq$ *?c*^{T} $*$ *?c* $\sqcap$ $--g$
proof $-$
have *16*: *w* $\sqcap$ $-$*?fp* $\leq$ *?c*^{T} $*$ *?c* $\sqcap$ $--g$
using *10* **by** (*simp add*: *le-infI1 minimum-spanning-tree-def spanning-tree-def*)

```
    have ?p^T ≤ w^T
      by (simp add: conv-isotone inf.sup-monoid.add-assoc)
    also have ... ≤ (?c^T * ?c ⊓ −−g)^T
      using 11 conv-order by simp
    also have ... = ?c^T * ?c ⊓ −−g
      using 2 14 conv-dist-comp conv-dist-inf by simp
    finally have 17: ?p^T ≤ ?c^T * ?c ⊓ −−g
      .
    have ?e ≤ ?c^T * ?c ⊓ ?g
      using 5 6 11 mst-subgraph-inv by auto
    thus ?thesis
      using 16 17 by simp
  qed
next
  show ?c ≤ r^T * ?w^⋆
  proof −
    have ?c ≤ r^T * w^⋆
      using 10 minimum-spanning-tree-def spanning-tree-def by simp
    also have ... ≤ r^T * ?w^⋆
      using 4 7 8 10 11 12 15 by (metis mst-reachable-inv)
    finally show ?thesis
      .
  qed
next
  show regular ?w
    using 15 by simp
qed
next
  have 18: ?f ⊔ ?p = ?fp
    using 2 8 epm-1 by fastforce
  have arc (w ⊓ −−v * −v^T ⊓ top * ?e * w^T^⋆)
    using 5 6 8 9 11 12 reachable-restrict arc-edge by auto
  hence 19: arc ?f
    using 2 by simp
  hence ?f = bot ⟶ top = bot
    by (metis mult-left-zero mult-right-zero)
  hence ?f ≠ bot
    using 1 le-bot by auto
  hence ?f ⊓ v * −v^T ⊓ ?g ≠ bot
    using 2 11 by (simp add: inf.absorb1 le-infI1)
  hence g ⊓ (?f ⊓ v * −v^T) ≠ bot
    using inf-commute pp-inf-bot-iff by simp
  hence 20: ?f ⊓ ?vcv ≠ bot
    by (simp add: inf-assoc inf-commute)
  hence 21: ?f ⊓ g = ?f ⊓ ?vcv
    using 2 by (simp add: inf-assoc inf-commute inf-left-commute)
  have 22: ?e ⊓ g = minarc ?vcv ⊓ ?vcv
    using 7 by (simp add: inf.absorb2 inf.assoc inf.commute)
  hence 23: sum (?e ⊓ g) ≤ sum (?f ⊓ g)
```

using *15 19 20 21* **by** (*simp add*: *minarc-min*)
have *?e* ≠ *bot*
using *20 comp-inf.semiring.mult-not-zero semiring.mult-not-zero* **by** *blast*
hence *24*: *?e* ⊓ *g* ≠ *bot*
using *22 minarc-meet-bot* **by** *auto*
have *sum* (*?w* ⊓ *g*) = *sum* (*w* ⊓ −*?fp* ⊓ *g*) + *sum* (*?p*^{T} ⊓ *g*) + *sum* (*?e* ⊓ *g*)
using *7 8 10* **by** (*metis sum-disjoint-3 epm-8 epm-9 epm-10 minimum-spanning-tree-def spanning-tree-def*)
also have ... = *sum* (((*w* ⊓ −*?fp*) ⊔ *?p*^{T}) ⊓ *g*) + *sum* (*?e* ⊓ *g*)
using *11* **by** (*metis epm-8 sum-disjoint*)
also have ... ≤ *sum* (((*w* ⊓ −*?fp*) ⊔ *?p*^{T}) ⊓ *g*) + *sum* (*?f* ⊓ *g*)
using *23 24* **by** (*simp add*: *sum-plus-right-isotone*)
also have ... = *sum* (*w* ⊓ −*?fp* ⊓ *g*) + *sum* (*?p*^{T} ⊓ *g*) + *sum* (*?f* ⊓ *g*)
using *11* **by** (*metis epm-8 sum-disjoint*)
also have ... = *sum* (*w* ⊓ −*?fp* ⊓ *g*) + *sum* (*?p* ⊓ *g*) + *sum* (*?f* ⊓ *g*)
using *13 sum-symmetric* **by** *auto*
also have ... = *sum* (((*w* ⊓ −*?fp*) ⊔ *?p* ⊔ *?f*) ⊓ *g*)
using *2 8* **by** (*metis sum-disjoint-3 epm-11 epm-12 epm-13*)
also have ... = *sum* (*w* ⊓ *g*)
using *2 8 15 18 epm-2* **by** *force*
finally have *sum* (*?w* ⊓ *g*) ≤ *sum* (*w* ⊓ *g*)
.
thus ∀ *u* . *spanning-tree u g r* ⟶ *sum* (*?w* ⊓ *g*) ≤ *sum* (*u* ⊓ *g*)
using *10 order-lesseq-imp minimum-spanning-tree-def* **by** *auto*
qed
next
show *?t* ≤ *?w*
using *4 8 10 mst-extends-new-tree* **by** *simp*
qed
qed
next
show *?n2* < *?n1*
using *1 prim-invariant-def prim-vc-2* **by** *auto*
qed
next
fix *t v*
let *?g* = −−*g*
assume *25*: *prim-invariant t v g r* ∧ *v* ∗ −*v*^{T} ⊓ *g* = *bot*
hence *26*: *regular v*
by (*metis prim-invariant-def prim-spanning-invariant-def spanning-tree-def prim-precondition-def regular-conv-closed regular-closed-star regular-mult-closed conv-involutive*)
from *25* **obtain** *w* **where** *27*: *minimum-spanning-tree w g r* ∧ *t* ≤ *w*
by (*metis prim-invariant-def*)
have *spanning-tree t g r*
using *25 prim-invariant-def prim-vc-3* **by** *blast*
hence *component g r* = *v*^{T}

by (*metis 25 prim-invariant-def span-tree-component prim-spanning-invariant-def spanning-tree-def*)
hence *28*: $w \leq v * v^T$
using *26 27* **by** (*simp add*: *minimum-spanning-tree-def spanning-tree-def inf-pp-commute*)
have *vector r* $\wedge$ *injective r* $\wedge$ *forest w*
using *25 27* **by** (*simp add*: *prim-invariant-def prim-spanning-invariant-def prim-precondition-def minimum-spanning-tree-def spanning-tree-def*)
hence $w = t$
using *25 27 28 prim-invariant-def prim-spanning-invariant-def mst-post* **by** *blast*
thus *minimum-spanning-tree t g r*
using *27* **by** *simp*
qed

end

end

4 Borůvka's Minimum Spanning Tree Algorithm

In this theory we prove partial correctness of Borůvka's minimum spanning tree algorithm.

theory *Boruvka*

imports
Aggregation-Algebras.M-Choose-Component
Relational-Disjoint-Set-Forests.Disjoint-Set-Forests
Kruskal

begin

4.1 General results

The proof is carried out in *m*-*k*-Stone-Kleene relation algebras. In this section we give results that hold more generally.

context *stone-kleene-relation-algebra*
begin

lemma *He-eq-He-THe-star*:
assumes *arc e*
and *equivalence H*
shows $H * e = H * e * (top * H * e)^\star$
proof −
let *?x* $= H * e$
have *1*: $H * e \leq H * e * (top * H * e)^\star$
using *comp-isotone star.circ-reflexive* **by** *fastforce*
have $H * e * (top * H * e)^\star = H * e * (top * e)^\star$

by (*metis assms(2) preorder-idempotent surjective-var*)
also have ... $\leq H * e * (1 \sqcup top * (e * top)^\star * e)$
by (*metis eq-refl star.circ-mult-1*)
also have ... $\leq H * e * (1 \sqcup top * top * e)$
by (*simp add*: *star.circ-left-top*)
also have ... $= H * e \sqcup H * e * top * e$
by (*simp add*: *mult.semigroup-axioms semiring.distrib-left semigroup.assoc*)
finally have *2*: $H * e * (top * H * e)^\star \leq H * e$
using *assms arc-top-arc mult-assoc* **by** *auto*
thus *?thesis*
using *1 2* **by** *simp*
qed

lemma *path-through-components*:
assumes *equivalence H*
and *arc e*
shows $(H * (d \sqcup e))^\star = (H * d)^\star \sqcup (H * d)^\star * H * e * (H * d)^\star$
proof −
have $H * e * (H * d)^\star * H * e \leq H * e * top * H * e$
by (*simp add*: *comp-isotone*)
also have ... $= H * e * top * e$
by (*metis assms(1) preorder-idempotent surjective-var mult-assoc*)
also have ... $= H * e$
using *assms(2) arc-top-arc mult-assoc* **by** *auto*
finally have *1*: $H * e * (H * d)^\star * H * e \leq H * e$
by *simp*
have $(H * (d \sqcup e))^\star = (H * d \sqcup H * e)^\star$
by (*simp add*: *comp-left-dist-sup*)
also have ... $= (H * d)^\star \sqcup (H * d)^\star * H * e * (H * d)^\star$
using *1 star-separate-3* **by** (*simp add*: *mult-assoc*)
finally show *?thesis*
by *simp*
qed

lemma *simplify-f*:
assumes *regular p*
and *regular e*
shows $(f \sqcap - e^T \sqcap - p) \sqcup (f \sqcap - e^T \sqcap p) \sqcup (f \sqcap - e^T \sqcap p)^T \sqcup (f \sqcap - e^T \sqcap - p)^T \sqcup e^T \sqcup e = f \sqcup f^T \sqcup e \sqcup e^T$
proof −
have $(f \sqcap - e^T \sqcap - p) \sqcup (f \sqcap - e^T \sqcap p) \sqcup (f \sqcap - e^T \sqcap p)^T \sqcup (f \sqcap - e^T \sqcap - p)^T \sqcup e^T \sqcup e$
$= (f \sqcap - e^T \sqcap - p) \sqcup (f \sqcap - e^T \sqcap p) \sqcup (f^T \sqcap - e \sqcap p^T) \sqcup (f^T \sqcap - e \sqcap - p^T) \sqcup e^T \sqcup e$
by (*simp add*: *conv-complement conv-dist-inf*)
also have ... $=$
$((f \sqcup (f \sqcap - e^T \sqcap p)) \sqcap (- e^T \sqcup (f \sqcap - e^T \sqcap p)) \sqcap (- p \sqcup (f \sqcap - e^T \sqcap p)))$
$\sqcup ((f^T \sqcup (f^T \sqcap - e \sqcap - p^T)) \sqcap (- e \sqcup (f^T \sqcap - e \sqcap - p^T)) \sqcap (p^T \sqcup (f^T \sqcap - e \sqcap - p^T)))$

$\sqcup\ e^T \sqcup e$
by (*metis sup-inf-distrib2 sup-assoc*)
also have ... =
$((f \sqcup f) \sqcap (f \sqcup - e^T) \sqcap (f \sqcup p) \sqcap (- e^T \sqcup f) \sqcap (- e^T \sqcup - e^T) \sqcap (- e^T \sqcup p) \sqcap (- p \sqcup f) \sqcap (- p \sqcup - e^T) \sqcap (- p \sqcup p))$
$\sqcup ((f^T \sqcup f^T) \sqcap (f^T \sqcup - e) \sqcap (f^T \sqcup - p^T) \sqcap (- e \sqcup f^T) \sqcap (- e \sqcup - e) \sqcap (- e \sqcup - p^T) \sqcap (p^T \sqcup f^T) \sqcap (p^T \sqcup - e) \sqcap (p^T \sqcup - p^T))$
$\sqcup\ e^T \sqcup e$
using *sup-inf-distrib1 sup-assoc inf-assoc sup-inf-distrib1* **by** *simp*
also have ... =
$((f \sqcup f) \sqcap (f \sqcup - e^T) \sqcap (f \sqcup p) \sqcap (f \sqcup - p) \sqcap (- e^T \sqcup f) \sqcap (- e^T \sqcup - e^T) \sqcap (- e^T \sqcup p) \sqcap (- e^T \sqcup - p) \sqcap (- p \sqcup p))$
$\sqcup ((f^T \sqcup f^T) \sqcap (f^T \sqcup - e) \sqcap (f^T \sqcup - p^T) \sqcap (- e \sqcup f^T) \sqcap (f^T \sqcup p^T) \sqcap (- e \sqcup - e) \sqcap (- e \sqcup - p^T) \sqcap (- e \sqcup p^T) \sqcap (p^T \sqcup - p^T))$
$\sqcup\ e^T \sqcup e$
by (*smt abel-semigroup.commute inf.abel-semigroup-axioms inf.left-commute sup.abel-semigroup-axioms*)
also have ... = $(f \sqcap - e^T \sqcap (- p \sqcup p)) \sqcup (f^T \sqcap - e \sqcap (p^T \sqcup - p^T)) \sqcup e^T \sqcup e$
by (*smt inf.sup-monoid.add-assoc inf.sup-monoid.add-commute inf-sup-absorb sup.idem*)
also have ... = $(f \sqcap - e^T) \sqcup (f^T \sqcap - e) \sqcup e^T \sqcup e$
by (*metis assms(1) conv-complement inf-top-right stone*)
also have ... = $(f \sqcup e^T) \sqcap (- e^T \sqcup e^T) \sqcup (f^T \sqcup e) \sqcap (- e \sqcup e)$
by (*metis sup.left-commute sup-assoc sup-inf-distrib2*)
finally show *?thesis*
by (*metis abel-semigroup.commute assms(2) conv-complement inf-top-right stone sup.abel-semigroup-axioms sup-assoc*)
qed

lemma *simplify-forest-components-f*:
assumes *regular p*
and *regular e*
and *injective* $(f \sqcap - e^T \sqcap - p \sqcup (f \sqcap - e^T \sqcap p)^T \sqcup e)$
and *injective f*
shows *forest-components* $((f \sqcap - e^T \sqcap - p) \sqcup (f \sqcap -e^T \sqcap p)^T \sqcup e) = (f \sqcup f^T \sqcup e \sqcup e^T)^\star$
proof −
have *forest-components* $((f \sqcap - e^T \sqcap - p) \sqcup (f \sqcap -e^T \sqcap p)^T \sqcup e) = wcc\ ((f \sqcap - e^T \sqcap - p) \sqcup (f \sqcap - e^T \sqcap p)^T \sqcup e)$
by (*simp add*: *assms(3) forest-components-wcc*)
also have ... = $((f \sqcap - e^T \sqcap - p) \sqcup (f \sqcap - e^T \sqcap p)^T \sqcup e \sqcup (f \sqcap - e^T \sqcap - p)^T \sqcup (f \sqcap - e^T \sqcap p) \sqcup e^T)^\star$
using *conv-dist-sup sup-assoc* **by** *auto*
also have ... = $((f \sqcap - e^T \sqcap - p) \sqcup (f \sqcap - e^T \sqcap p) \sqcup (f \sqcap - e^T \sqcap p)^T \sqcup (f \sqcap - e^T \sqcap - p)^T \sqcup e^T \sqcup e)^\star$
using *sup-assoc sup-commute* **by** *auto*
also have ... = $(f \sqcup f^T \sqcup e \sqcup e^T)^\star$
using *assms(1, 2, 3, 4) simplify-f* **by** *auto*
finally show *?thesis*

by *simp*
qed

lemma *components-disj-increasing*:
assumes *regular p*
and *regular e*
and *injective* $(f \sqcap - e^T \sqcap - p \sqcup (f \sqcap - e^T \sqcap p)^T \sqcup e)$
and *injective f*
shows *forest-components f* $\leq$ *forest-components* $(f \sqcap - e^T \sqcap - p \sqcup (f \sqcap - e^T \sqcap p)^T \sqcup e)$
proof −
have *1*: *forest-components* $((f \sqcap - e^T \sqcap - p) \sqcup (f \sqcap -e^T \sqcap p)^T \sqcup e) = (f \sqcup f^T \sqcup e \sqcup e^T)^\star$
using *simplify-forest-components-f assms*(*1*, *2*, *3*, *4*) **by** *blast*
have *forest-components f* = *wcc f*
by (*simp add*: *assms*(*4*) *forest-components-wcc*)
also have ... $\leq (f \sqcup f^T \sqcup e^T \sqcup e)^\star$
by (*simp add*: *le-supI2 star-isotone sup-commute*)
finally show *?thesis*
using *1 sup.left-commute sup-commute* **by** *simp*
qed

lemma *fch-equivalence*:
assumes *forest h*
shows *equivalence* (*forest-components h*)
by (*simp add*: *assms forest-components-equivalence*)

lemma *forest-modulo-equivalence-path-split-1*:
assumes *arc a*
and *equivalence H*
shows $(H * d)^\star * H * a * top = (H * (d \sqcap - a))^\star * H * a * top$
proof −
let *?H* = *H*
let *?x* = *?H* $*$ $(d \sqcap -a)$
let *?y* = *?H* $*$ *a*
let *?a* = *?H* $*$ *a* $*$ *top*
let *?d* = *?H* $*$ *d*
have *1*: $?d^\star * ?a \leq ?x^\star * ?a$
proof −
have $?x^\star * ?y * ?x^\star * ?a \leq ?x^\star * ?a * ?a$
by (*smt mult-left-isotone star.circ-right-top top-right-mult-increasing mult-assoc*)
also have ... = $?x^\star * ?a * a * top$
by (*metis ex231e mult-assoc*)
also have ... = $?x^\star * ?a$
by (*simp add*: *assms*(*1*) *mult-assoc*)
finally have *11*: $?x^\star * ?y * ?x^\star * ?a \leq ?x^\star * ?a$
by *simp*
have $?d^\star * ?a = (?H * (d \sqcap a) \sqcup ?H * (d \sqcap -a))^\star * ?a$

```
    proof −
      have 12: regular a
        using assms(1) arc-regular by simp
      have ?H ∗ ((d ⊓ a) ⊔ (d ⊓ − a)) = ?H ∗ (d ⊓ top)
        using 12 by (metis inf-top-right maddux-3-11-pp)
      thus ?thesis
        using mult-left-dist-sup by auto
    qed
    also have ... ≤ (?y ⊔ ?x)^⋆ ∗ ?a
      by (metis comp-inf.coreflexive-idempotent comp-isotone inf.cobounded1
inf.sup-monoid.add-commute semiring.add-mono star-isotone top.extremum)
    also have ... = (?x ⊔ ?y)^⋆ ∗ ?a
      by (simp add: sup-commute mult-assoc)
    also have ... = ?x^⋆ ∗ ?a ⊔ (?x^⋆ ∗ ?y ∗ (?x^⋆ ∗ ?y)^⋆ ∗ ?x^⋆) ∗ ?a
      by (smt mult-right-dist-sup star.circ-sup-9 star.circ-unfold-sum mult-assoc)
    also have ... ≤ ?x^⋆ ∗ ?a ⊔ (?x^⋆ ∗ ?y ∗ (top ∗ ?y)^⋆ ∗ ?x^⋆) ∗ ?a
    proof −
      have (?x^⋆ ∗ ?y)^⋆ ≤ (top ∗ ?y)^⋆
        by (simp add: mult-left-isotone star-isotone)
      thus ?thesis
        by (metis comp-inf.coreflexive-idempotent comp-inf.transitive-star eq-refl
mult-left-dist-sup top.extremum mult-assoc)
    qed
    also have ... = ?x^⋆ ∗ ?a ⊔ (?x^⋆ ∗ ?y ∗ ?x^⋆) ∗ ?a
      using assms(1, 2) He-eq-He-THe-star arc-regular mult-assoc by auto
    finally have 13: (?H ∗ d)^⋆ ∗ ?a ≤ ?x^⋆ ∗ ?a ⊔ ?x^⋆ ∗ ?y ∗ ?x^⋆ ∗ ?a
      by (simp add: mult-assoc)
    have 14: ?x^⋆ ∗ ?y ∗ ?x^⋆ ∗ ?a ≤ ?x^⋆ ∗ ?a
      using 11 mult-assoc by auto
    thus ?thesis
      using 13 14 sup.absorb1 by auto
  qed
  have 2: ?d^⋆ ∗ ?a ≥ ?x^⋆ ∗?a
    by (simp add: comp-isotone star-isotone)
  thus ?thesis
    using 1 2 order.antisym mult-assoc by simp
qed

lemma dTransHd-le-1:
  assumes equivalence H
    and univalent (H ∗ d)
  shows d^T ∗ H ∗ d ≤ 1
proof −
  have d^T ∗ H^T ∗ H ∗ d ≤ 1
    using assms(2) conv-dist-comp mult-assoc by auto
  thus ?thesis
    using assms(1) mult-assoc by (simp add: preorder-idempotent)
qed
```

lemma *HcompaT-le-compHaT*:
assumes *equivalence H*
and *injective* $(a * top)$
shows $-H * a * top \leq -(H * a * top)$
proof −
have $a * top * a^T \leq 1$
by (*metis assms*(2) *conv-dist-comp symmetric-top-closed vector-top-closed mult-assoc*)
hence $a * top * a^T * H \leq H$
using *assms*(1) *comp-isotone order-trans* **by** *blast*
hence $a * top * top * a^T * H \leq H$
by (*simp add*: *vector-mult-closed*)
hence $a * top * (H * a * top)^T \leq H$
by (*metis assms*(1) *conv-dist-comp symmetric-top-closed vector-top-closed mult-assoc*)
thus *?thesis*
using *assms*(2) *comp-injective-below-complement mult-assoc* **by** *auto*
qed

4.2 Forests modulo an equivalence

In the graphical interpretation, the arcs of d are directed towards the root(s) of the *forest-modulo-equivalence*.

definition *forest-modulo-equivalence x d* ≡ *equivalence x* ∧ *univalent* $(x * d)$ ∧ $x \sqcap d * d^T \leq 1 \wedge (x * d)^+ \sqcap x \leq bot$

definition *forest-modulo-equivalence-path a b H d* ≡ *arc a* ∧ *arc b* ∧ $a^T * top \leq (H * d)^\star * H * b * top$

lemma *d-separates-forest-modulo-equivalence-1*:
assumes *forest-modulo-equivalence x d*
shows $x * d \leq -x$
proof −
have $x * d \leq (x * d)^+$
using *star.circ-mult-increasing* **by** *simp*
also have ... $\leq -x$
using *assms*(1) *forest-modulo-equivalence-def le-bot pseudo-complement* **by** *blast*
finally show *?thesis*
by *simp*
qed

lemma *d-separates-forest-modulo-equivalence-2*:
shows *forest-modulo-equivalence x d* ⟹ $d * x \leq -x$
using *forest-modulo-equivalence-def schroeder-6-p d-separates-forest-modulo-equivalence-1* **by** *metis*

lemma *d-separates-forest-modulo-equivalence-3*:
assumes *forest-modulo-equivalence x d*

shows $d \leq -x$
proof −
have *1* $\leq x$
using *assms(1) forest-modulo-equivalence-def* **by** *auto*
then have $d \leq x * d$
using *mult-left-isotone* **by** *fastforce*
also have ... $\leq (x * d)^+$
using *star.circ-mult-increasing* **by** *simp*
also have ... $\leq -x$
using *assms(1) forest-modulo-equivalence-def le-bot pseudo-complement* **by** *blast*
finally show *?thesis*
by *simp*
qed

lemma *d-separates-forest-modulo-equivalence-4*:
shows *forest-modulo-equivalence x d* $\Longrightarrow d^T \leq -x$
using *d-separates-forest-modulo-equivalence-3 forest-modulo-equivalence-def conv-isotone symmetric-complement-closed* **by** *metis*

lemma *d-separates-forest-modulo-equivalence-5*:
shows *forest-modulo-equivalence x d* $\Longrightarrow d \sqcup d^T \leq -x$
using *d-separates-forest-modulo-equivalence-3 d-separates-forest-modulo-equivalence-4 sup-least* **by** *blast*

lemma *d-separates-forest-modulo-equivalence-6*:
shows *forest-modulo-equivalence x d* $\Longrightarrow d * x \sqcup x * d \leq -x$
using *d-separates-forest-modulo-equivalence-1 d-separates-forest-modulo-equivalence-2 sup-least* **by** *blast*

lemma *d-separates-forest-modulo-equivalence-7*:
shows *forest-modulo-equivalence x d* $\Longrightarrow x * (d \sqcup d^T) * x \leq -x$
using *d-separates-forest-modulo-equivalence-5 forest-modulo-equivalence-def inf.sup-monoid.add-commute preorder-idempotent pseudo-complement triple-schroeder-p* **by** *metis*

lemma *d-separates-forest-modulo-equivalence-8*:
shows *forest-modulo-equivalence x d* $\Longrightarrow (d * x)^T \leq -x$
using *d-separates-forest-modulo-equivalence-2 forest-modulo-equivalence-def conv-isotone symmetric-complement-closed* **by** *metis*

lemma *d-separates-forest-modulo-equivalence-9*:
shows *forest-modulo-equivalence x d* $\Longrightarrow (x * d)^T \leq -x$
by (*metis d-separates-forest-modulo-equivalence-1 forest-modulo-equivalence-def conv-isotone symmetric-complement-closed*)

lemma *d-separates-forest-modulo-equivalence-10*:
shows *forest-modulo-equivalence x d* $\Longrightarrow (d * x)^+ \leq -x$
using *forest-modulo-equivalence-def le-bot pseudo-complement schroeder-5-p*

star-slide mult-assoc **by** *metis*

lemma *d-separates-forest-modulo-equivalence-11*:
shows *forest-modulo-equivalence x d* $\Longrightarrow (x * d)^+ \leq -x$
using *forest-modulo-equivalence-def le-bot pseudo-complement* **by** *blast*

lemma *d-separates-forest-modulo-equivalence-12*:
shows *forest-modulo-equivalence x d* $\Longrightarrow (d * x)^{T+} \leq -x$
using *d-separates-forest-modulo-equivalence-10 forest-modulo-equivalence-def conv-isotone conv-plus-commute symmetric-complement-closed* **by** *metis*

lemma *d-separates-x-in-forest-13*:
shows *forest-modulo-equivalence x d* $\Longrightarrow (x * d)^{T+} \leq -x$
using *d-separates-forest-modulo-equivalence-11 forest-modulo-equivalence-def conv-isotone conv-plus-commute symmetric-complement-closed* **by** *metis*

lemma *irreflexive-d-in-forest-modulo-equivalence*:
shows *forest-modulo-equivalence x d* $\Longrightarrow$ *irreflexive d*
by (*metis d-separates-forest-modulo-equivalence-3 forest-modulo-equivalence-def inf.cobounded2 inf.left-commute inf.orderE pseudo-complement*)

lemma *univalent-d-in-forest-modulo-equivalence*:
assumes *forest-modulo-equivalence x d*
shows *univalent d*
proof −
have $d^T * d \leq d^T * x^T * x * d$
using *assms*(*1*) *forest-modulo-equivalence-def comp-isotone comp-right-one mult-sub-right-one* **by** *metis*
also have ... $\leq$ *1*
using *assms*(*1*) *forest-modulo-equivalence-def comp-associative conv-dist-comp* **by** *auto*
finally show *?thesis*
by *simp*
qed

lemma *acyclic-d-in-forest-modulo-equivalence*:
assumes *forest-modulo-equivalence x d*
shows *acyclic d*
proof −
have $d^\star \leq (x * d)^\star$
using *assms*(*1*) *forest-modulo-equivalence-def mult-left-isotone star.circ-circ-mult star.circ-circ-mult-1 star.circ-extra-circ star.left-plus-circ star-involutive star-isotone star-one star-slide mult-assoc* **by** *metis*
then have $d * d^\star \leq d * (x * d)^\star$
using *mult-right-isotone* **by** *blast*
also have ... $\leq x * d * (x * d)^\star$
using *assms*(*1*) *forest-modulo-equivalence-def eq-refl inf.order-trans mult-isotone star.circ-circ-mult-1 star-involutive star-one star-outer-increasing mult-assoc* **by** *metis*

also have ... $\leq -x$
using *assms d-separates-forest-modulo-equivalence-11* **by** *blast*
also have ... ≤ -1
using *assms(1) forest-modulo-equivalence-def p-antitone* **by** *blast*
finally show *?thesis*
by *simp*
qed

lemma *acyclic-dt-d-in-forest-modulo-equivalence*:
shows *forest-modulo-equivalence x d* $\Longrightarrow$ *acyclic* (d^T)
using *acyclic-d-in-forest-modulo-equivalence conv-plus-commute irreflexive-conv-closed* **by** *fastforce*

lemma *dt-forest-modulo-equivalence-forest*:
shows *forest-modulo-equivalence x d* $\Longrightarrow$ *forest* (d^T)
using *acyclic-dt-d-in-forest-modulo-equivalence univalent-d-in-forest-modulo-equivalence* **by** *simp*

lemma *var-forest-modulo-equivalence-axiom*:
shows *forest-modulo-equivalence x d* $\Longrightarrow$ $d^T * x * d \leq 1$
using *forest-modulo-equivalence-def comp-associative conv-dist-comp preorder-idempotent* **by** *metis*

lemma *forest-modulo-equivalence-wcc*:
assumes *forest-modulo-equivalence x d*
shows $(x * d)^\star * (x * d)^{T\star} = ((x * d) \sqcup (x * d)^T)^\star$
using *assms(1) forest-modulo-equivalence-def fc-wcc* **by** *force*

lemma *forest-modulo-equivalence-direction-1*:
assumes *forest-modulo-equivalence x d*
shows $(x * d)^\star \sqcap (x * d)^T = bot$
using *assms(1) d-separates-forest-modulo-equivalence-11 forest-modulo-equivalence-def acyclic-star-below-complement-1 order-lesseq-imp p-antitone-iff* **by** *meson*

lemma *forest-modulo-equivalence-direction-2*:
assumes *forest-modulo-equivalence x d*
shows $(x * d)^{T\star} \sqcap (x * d) \leq bot$
using *assms(1) forest-modulo-equivalence-direction-1 comp-inf.idempotent-bot-closed conv-inf-bot-iff conv-star-commute inf.sup-left-divisibility* **by** *metis*

lemma *forest-modulo-equivalence-separate*:
assumes *forest-modulo-equivalence x d*
shows $(x * d)^\star * (x * d)^{T\star} \sqcap (x * d)^T * (x * d) \leq 1$
proof −
have $(x * d)^\star \sqcap (x * d)^T * (x * d) = (1 \sqcup (x * d)^+) \sqcap (x * d)^T * (x * d)$
using *star-left-unfold-equal* **by** *simp*
also have ... $= (1 \sqcap (x * d)^T * (x * d)) \sqcup ((x * d)^+ \sqcap (x * d)^T * (x * d))$

using *comp-inf.semiring.distrib-right* **by** *simp*
also have ... $\leq$ *1* $\sqcup$ $((x * d)^+ \sqcap (x * d)^T * (x * d))$
using *inf.cobounded1 semiring.add-right-mono* **by** *blast*
also have ... = *1* $\sqcup$ $((x * d)^\star \sqcap (x * d)^T) * (x * d)$
using *assms(1) forest-modulo-equivalence-def*
forest-modulo-equivalence-direction-1 comp-inf.semiring.mult-zero-right
inf.sup-left-divisibility le-infI2 semiring.mult-not-zero sup.orderE **by** *metis*
also have ... $\leq$ *1* $\sqcup$ *bot*
using *assms(1) forest-modulo-equivalence-direction-1* **by** *simp*
finally have *2*: $(x * d)^\star \sqcap (x * d)^T * (x * d) \leq$ *1*
by *simp*
then have *3*: $(x * d)^{T\star} \sqcap (x * d)^T * (x * d) \leq$ *1*
using *assms(1) forest-modulo-equivalence-def conv-dist-comp conv-dist-inf*
conv-involutive conv-star-commute coreflexive-symmetric **by** *metis*
have $((x * d)^\star \sqcup (x * d)^{T\star}) \sqcap ((x * d)^T * (x * d)) \leq$ *1*
using *2 3 inf-sup-distrib2* **by** *simp*
thus *?thesis*
using *assms(1) le-infI2 forest-modulo-equivalence-def* **by** *blast*
qed

lemma *forest-modulo-equivalence-path-trans-closure*:
assumes *forest-modulo-equivalence x d*
shows $(x * d^T)^+ * x * d * x * d^T \leq (x * d^T)^+$
proof −
have $(x * d^T)^+ * x * d * x * d^T = (x * d^T)^\star * x * d^T * x * d * x * d^T$
using *comp-associative star.circ-plus-same* **by** *metis*
also have ... $\leq (x * d^T)^\star * x * 1 * x * d^T$
using *assms(1) forest-modulo-equivalence-def*
var-forest-modulo-equivalence-axiom comp-associative mult-left-isotone
mult-right-isotone **by** *metis*
also have ... $\leq (x * d^T)^\star * x * d^T$
using *assms(1) forest-modulo-equivalence-def* **by** (*simp add*:
preorder-idempotent mult-assoc)
finally show *?thesis*
using *star.circ-plus-same mult-assoc* **by** *simp*
qed

The *forest-modulo-equivalence* structure is preserved if d is decreased.

lemma *forest-modulo-equivalence-decrease-d*:
assumes *forest-modulo-equivalence x d*
shows *forest-modulo-equivalence x* $(d \sqcap c)$
proof (*unfold forest-modulo-equivalence-def*, *intro conjI*)
show *reflexive x*
using *assms(1) forest-modulo-equivalence-def* **by** *blast*
next
show *transitive x*
using *assms(1) forest-modulo-equivalence-def* **by** *blast*
next
show *symmetric x*

using *assms(1) forest-modulo-equivalence-def* **by** *blast*
next
show *univalent* $(x * (d \sqcap c))$
proof −
have $(x * (d \sqcap c))^T * x * (d \sqcap c) \leq (x * d)^T * x * d$
using *conv-order mult-isotone* **by** *auto*
also have ... $\leq$ *1*
using *assms(1) forest-modulo-equivalence-def mult-assoc* **by** *auto*
finally show *?thesis*
using *mult-assoc* **by** *auto*
qed
next
show *coreflexive* $(x \sqcap ((d \sqcap c) * (d \sqcap c)^T))$
proof −
have $x \sqcap (d \sqcap c) * (d \sqcap c)^T \leq x \sqcap d * d^T$
using *conv-dist-inf inf.sup-right-isotone mult-isotone* **by** *auto*
thus *?thesis*
using *assms(1) forest-modulo-equivalence-def order-lesseq-imp* **by** *blast*
qed
next
show $(x * (d \sqcap c))^+ \sqcap x \leq bot$
proof −
have $(x * (d \sqcap c))^+ \leq (x * d)^+$
using *comp-isotone star-isotone* **by** *simp*
thus *?thesis*
using *assms d-separates-forest-modulo-equivalence-11 dual-order.eq-iff dual-order.trans pseudo-complement* **by** *blast*
qed
qed

lemma *expand-forest-modulo-equivalence*:
assumes *forest-modulo-equivalence H d*
shows $(d^T * H)^\star * (H * d)^\star = (d^T * H)^\star \sqcup (H * d)^\star$
proof −
have $(H * d)^T * H * d \leq 1$
using *assms forest-modulo-equivalence-def mult-assoc* **by** *auto*
hence $d^T * H * H * d \leq 1$
using *assms forest-modulo-equivalence-def conv-dist-comp* **by** *auto*
thus *?thesis*
by (*simp add*: *cancel-separate-eq comp-associative*)
qed

lemma *forest-modulo-equivalence-path-bot*:
assumes *arc a*
and $a \leq d$
and *forest-modulo-equivalence H d*
shows $(d \sqcap - a)^T * (H * a * top) \leq bot$
proof −
have *1*: $d^T * H * d \leq 1$

using *assms*(*3*) *forest-modulo-equivalence-def dTransHd-le-1* **by** *blast*
hence $d * - 1 * d^T \leq - H$
using *triple-schroeder-p* **by** *force*
hence $d * - 1 * d^T \leq 1 \sqcup - H$
by (*simp add*: *le-supI2*)
hence $d * d^T \sqcup d * - 1 * d^T \leq 1 \sqcup - H$
by (*metis assms*(*3*) *forest-modulo-equivalence-def inf-commute regular-one-closed shunting-p le-supI*)
hence $d * 1 * d^T \sqcup d * - 1 * d^T \leq 1 \sqcup - H$
by *simp*
hence $d * (1 \sqcup - 1) * d^T \leq 1 \sqcup - H$
using *comp-associative mult-right-dist-sup* **by** (*simp add*: *mult-left-dist-sup*)
hence $d * top * d^T \leq 1 \sqcup - H$
using *regular-complement-top* **by** *auto*
hence $d * top * a^T \leq 1 \sqcup - H$
using *assms*(*2*) *conv-isotone dual-order.trans mult-right-isotone* **by** *blast*
hence $d * (a * top)^T \leq 1 \sqcup - H$
by (*simp add*: *comp-associative conv-dist-comp*)
hence $d \leq (1 \sqcup - H) * (a * top)$
by (*simp add*: *assms*(*1*) *shunt-bijective*)
hence $d \leq a * top \sqcup - H * a * top$
by (*simp add*: *comp-associative mult-right-dist-sup*)
also have $... \leq a * top \sqcup - (H * a * top)$
using *assms*(*1*, *3*) *HcompaT-le-compHaT forest-modulo-equivalence-def sup-right-isotone* **by** *auto*
finally have $d \leq a * top \sqcup - (H * a * top)$
by *simp*
hence $d \sqcap - - (H * a * top) \leq a * top$
using *shunting-var-p* **by** *auto*
hence *2*:$d \sqcap H * a * top \leq a * top$
using *inf.sup-right-isotone order.trans pp-increasing* **by** *blast*
have *3*:$d \sqcap H * a * top \leq top * a$
proof −
have $d \sqcap H * a * top \leq (H * a \sqcap d * top^T) * (top \sqcap (H * a)^T * d)$
by (*metis dedekind inf-commute*)
also have $... = d * top \sqcap H * a * a^T * H^T * d$
by (*simp add*: *conv-dist-comp inf-vector-comp mult-assoc*)
also have $... \leq d * top \sqcap H * a * d^T * H^T * d$
using *assms*(*2*) *mult-right-isotone mult-left-isotone conv-isotone inf.sup-right-isotone* **by** *auto*
also have $... = d * top \sqcap H * a * d^T * H * d$
using *assms*(*3*) *forest-modulo-equivalence-def* **by** *auto*
also have $... \leq d * top \sqcap H * a * 1$
using *1* **by** (*metis inf.sup-right-isotone mult-right-isotone mult-assoc*)
also have $... \leq H * a$
by *simp*
also have $... \leq top * a$
by (*simp add*: *mult-left-isotone*)
finally have $d \sqcap H * a * top \leq top * a$

by *simp*
thus *?thesis*
by *simp*
qed
have $d \sqcap H * a * top \leq a * top \sqcap top * a$
using *2 3* **by** *simp*
also have $... = a * top * top * a$
by (*metis comp-associative comp-inf.star.circ-decompose-9 comp-inf.star-star-absorb comp-inf-covector vector-inf-comp vector-top-closed*)
also have $... = a * top * a$
by (*simp add*: *vector-mult-closed*)
finally have *4*:$d \sqcap H * a * top \leq a$
by (*simp add*: *assms(1) arc-top-arc*)
hence $d \sqcap - a \leq -(H * a * top)$
using *assms(1) arc-regular p-shunting-swap* **by** *fastforce*
hence $(d \sqcap - a) * top \leq -(H * a * top)$
using *mult.semigroup-axioms p-antitone-iff schroeder-4-p semigroup.assoc* **by** *fastforce*
thus *?thesis*
by (*simp add*: *schroeder-3-p*)
qed

lemma *forest-modulo-equivalence-path-split-2*:
assumes *arc a*
and $a \leq d$
and *forest-modulo-equivalence H d*
shows $(H * (d \sqcap - a))^\star * H * a * top = (H * ((d \sqcap - a) \sqcup (d \sqcap - a)^T))^\star * H * a * top$
proof −
let *?lhs* $= (H * (d \sqcap - a))^\star * H * a * top$
have *1*: $d^T * H * d \leq 1$
using *assms(3) forest-modulo-equivalence-def dTransHd-le-1* **by** *blast*
have *2*: $H * a * top \leq$ *?lhs*
by (*metis le-iff-sup star.circ-loop-fixpoint star.circ-transitive-equal star-involutive sup-commute mult-assoc*)
have $(d \sqcap - a)^T * (H * (d \sqcap - a))^\star * (H * a * top) = (d \sqcap - a)^T * H * (d \sqcap - a) * (H * (d \sqcap - a))^\star * (H * a * top)$
proof −
have $(d \sqcap - a)^T * (H * (d \sqcap - a))^\star * (H * a * top) = (d \sqcap - a)^T * (1 \sqcup H * (d \sqcap - a) * (H * (d \sqcap - a))^\star) * (H * a * top)$
by (*simp add*: *star-left-unfold-equal*)
also have $... = (d \sqcap - a)^T * H * a * top \sqcup (d \sqcap - a)^T * H * (d \sqcap - a) * (H * (d \sqcap - a))^\star * (H * a * top)$
by (*smt mult-left-dist-sup star.circ-loop-fixpoint star.circ-mult-1 star-slide sup-commute mult-assoc*)
also have $... = bot \sqcup (d \sqcap - a)^T * H * (d \sqcap - a) * (H * (d \sqcap - a))^\star * (H * a * top)$
by (*metis assms(1, 2, 3) forest-modulo-equivalence-path-bot mult-assoc le-bot*)

thus *?thesis*
by (*simp add*: *calculation*)
qed
also have ... $\leq d^T * H * d * (H * (d \sqcap - a))^\star * (H * a * top)$
using *conv-isotone inf.cobounded1 mult-isotone* **by** *auto*
also have ... $\leq 1 * (H * (d \sqcap - a))^\star * (H * a * top)$
using *1* **by** (*metis le-iff-sup mult-right-dist-sup*)
finally have *3*: $(d \sqcap - a)^T * (H * (d \sqcap - a))^\star * (H * a * top) \leq$ *?lhs*
using *mult-assoc* **by** *auto*
hence *4*: $H * (d \sqcap - a)^T * (H * (d \sqcap - a))^\star * (H * a * top) \leq$ *?lhs*
proof −
have $H * (d \sqcap - a)^T * (H * (d \sqcap - a))^\star * (H * a * top) \leq H * (H * (d \sqcap - a))^\star * H * a * top$
using *3 mult-right-isotone mult-assoc* **by** *auto*
also have ... $= H * H * ((d \sqcap - a) * H)^\star * H * a * top$
by (*metis assms*(*3*) *forest-modulo-equivalence-def star-slide mult-assoc preorder-idempotent*)
also have ... $= H * ((d \sqcap - a) * H)^\star * H * a * top$
using *assms*(*3*) *forest-modulo-equivalence-def preorder-idempotent* **by** *fastforce*
finally show *?thesis*
by (*metis assms*(*3*) *forest-modulo-equivalence-def preorder-idempotent star-slide mult-assoc*)
qed
have *5*: $(H * (d \sqcap - a) \sqcup H * (d \sqcap - a)^T) * (H * (d \sqcap - a))^\star * H * a * top \leq$ *?lhs*
proof −
have *51*: $H * (d \sqcap - a) * (H * (d \sqcap - a))^\star * H * a * top \leq (H * (d \sqcap - a))^\star * H * a * top$
using *star.left-plus-below-circ mult-left-isotone* **by** *simp*
have *52*: $(H * (d \sqcap - a) \sqcup H * (d \sqcap - a)^T) * (H * (d \sqcap - a))^\star * H * a * top = H * (d \sqcap - a) * (H * (d \sqcap - a))^\star * H * a * top \sqcup H * (d \sqcap - a)^T * (H * (d \sqcap - a))^\star * H * a * top$
using *mult-right-dist-sup* **by** *auto*
hence ... $\leq (H * (d \sqcap - a))^\star * H * a * top \sqcup H * (d \sqcap - a)^T * (H * (d \sqcap - a))^\star * H * a * top$
using *star.left-plus-below-circ mult-left-isotone sup-left-isotone* **by** *auto*
thus *?thesis*
using *4 51 52 mult-assoc* **by** *auto*
qed
hence $(H * (d \sqcap - a) \sqcup H * (d \sqcap - a)^T)^\star * H * a * top \leq$ *?lhs*
proof −
have $(H * (d \sqcap - a) \sqcup H * (d \sqcap - a)^T)^\star * (H * (d \sqcap - a))^\star * H * a * top \leq$ *?lhs*
using *5 star-left-induct-mult-iff mult-assoc* **by** *auto*
thus *?thesis*
using *star.circ-decompose-11 star-decompose-1* **by** *auto*
qed
hence *6*: $(H * ((d \sqcap - a) \sqcup (d \sqcap - a)^T))^\star * H * a * top \leq$ *?lhs*

using *mult-left-dist-sup* **by** *auto*
have *7*: $(H * (d \sqcap - a))^\star * H * a * top \leq (H * ((d \sqcap - a) \sqcup (d \sqcap - a)^T))^\star * H * a * top$
by (*simp add*: *mult-left-isotone semiring.distrib-left star-isotone*)
thus *?thesis*
using *6 7* **by** (*simp add*: *mult-assoc*)
qed

end

4.3 An operation to select components

This section has been moved to theories *Stone-Relation-Algebras.Choose-Component* and *Aggregation-Algebras.M-Choose-Component*.

4.4 m-k-Stone-Kleene relation algebras

m-k-Stone-Kleene relation algebras are an extension of *m*-Kleene algebras where the *choose-component* operation has been added.

context *m-kleene-algebra-choose-component*
begin

A *selected-edge* is a minimum-weight edge whose source is in a component, with respect to h, j and g, and whose target is not in that component.

abbreviation *selected-edge h j g* $\equiv$ *minarc* (*choose-component* (*forest-components h*) *j* $*$ $-$ *choose-component* (*forest-components h*) $j^T \sqcap g$)

A *path* is any sequence of edges in the forest, f, of the graph, g, backwards from the target of the *selected-edge* to a root in f.

abbreviation *path f h j g* $\equiv$ *top* $*$ *selected-edge h j g* $*$ $(f \sqcap -$ *selected-edge h j* $g^T)^{T\star}$

definition *boruvka-outer-invariant f g* $\equiv$
symmetric g
$\wedge$ *forest f*
$\wedge$ $f \leq --g$
$\wedge$ *regular f*
$\wedge$ $(\exists w$. *minimum-spanning-forest w g* $\wedge$ $f \leq w \sqcup w^T)$

definition *boruvka-inner-invariant j f h g d* $\equiv$
boruvka-outer-invariant f g
$\wedge$ $g \neq bot$
$\wedge$ *regular d*
$\wedge$ *regular j* $\wedge$ *vector j*
$\wedge$ *regular h* $\wedge$ *forest h*
$\wedge$ *forest-components h* $*$ $j = j$
$\wedge$ *forest-modulo-equivalence* (*forest-components h*) *d*
$\wedge$ $d * top \leq - j$

$\wedge\ f \sqcup f^T = h \sqcup h^T \sqcup d \sqcup d^T$
$\wedge\ (\forall\ a\ b\ .\ \mathit{forest\text{-}modulo\text{-}equivalence\text{-}path}\ a\ b\ (\mathit{forest\text{-}components}\ h)\ d \wedge a \leq -(\mathit{forest\text{-}components}\ h) \sqcap --\ g \wedge b \leq d \longrightarrow \mathit{sum}(b \sqcap g) \leq \mathit{sum}(a \sqcap g))$

lemma *F-is-H-and-d*:
assumes $f \sqcup f^T = h \sqcup h^T \sqcup d \sqcup d^T$
and *injective f*
and *injective h*
shows $\mathit{forest\text{-}components}\ f = (\mathit{forest\text{-}components}\ h * (d \sqcup d^T))^\star * \mathit{forest\text{-}components}\ h$
proof −
have $\mathit{forest\text{-}components}\ f = (f \sqcup f^T)^\star$
using *assms*(2) *cancel-separate-1* **by** *simp*
also have ... $= (h \sqcup h^T \sqcup d \sqcup d^T)^\star$
using *assms*(1) **by** *auto*
also have ... $= ((h \sqcup h^T)^\star * (d \sqcup d^T))^\star * (h \sqcup h^T)^\star$
using *star.circ-sup-9 sup-assoc* **by** *metis*
also have ... $= (\mathit{forest\text{-}components}\ h * (d \sqcup d^T))^\star * \mathit{forest\text{-}components}\ h$
using *assms*(3) *forest-components-wcc* **by** *simp*
finally show *?thesis*
by *simp*
qed

lemma *H-below-F*:
assumes $f \sqcup f^T = h \sqcup h^T \sqcup d \sqcup d^T$
and *injective f*
and *injective h*
shows $\mathit{forest\text{-}components}\ h \leq \mathit{forest\text{-}components}\ f$
using *assms*(1, 2, 3) *cancel-separate-1 dual-order.trans star.circ-sub-dist* **by** *metis*

lemma *H-below-regular-g*:
assumes $f \sqcup f^T = h \sqcup h^T \sqcup d \sqcup d^T$
and $f \leq --g$
and *symmetric g*
shows $h \leq --g$
proof −
have $h \leq f \sqcup f^T$
using *assms*(1) *sup-assoc* **by** *simp*
also have ... $\leq --g$
using *assms*(2, 3) *conv-complement conv-isotone* **by** *fastforce*
finally show *?thesis*
using *order-trans* **by** *blast*
qed

lemma *expression-equivalent-without-e-complement*:
assumes $\mathit{selected\text{-}edge}\ h\ j\ g \leq -\ \mathit{forest\text{-}components}\ f$
shows $f \sqcap -(\mathit{selected\text{-}edge}\ h\ j\ g)^T \sqcap -(\mathit{path}\ f\ h\ j\ g) \sqcup (f \sqcap -(\mathit{selected\text{-}edge}\ h\ j\ g)^T \sqcap (\mathit{path}\ f\ h\ j\ g))^T \sqcup (\mathit{selected\text{-}edge}\ h\ j\ g)$

$= f \sqcap - (path\ f\ h\ j\ g) \sqcup (f \sqcap (path\ f\ h\ j\ g))^T \sqcup (selected\text{-}edge\ h\ j\ g)$
proof −
let *?p* = *path f h j g*
let *?e* = *selected-edge h j g*
let *?F* = *forest-components f*
have *1*: *?e* ≤ − *?F*
by (*simp add*: *assms*)
have $f^T \leq$ *?F*
by (*metis conv-dist-comp conv-involutive conv-order conv-star-commute forest-components-increasing*)
hence − *?F* ≤ − f^T
using *p-antitone* **by** *auto*
hence *?e* ≤ − f^T
using *1 dual-order.trans* **by** *blast*
hence $f^T \leq$ − *?e*
by (*simp add*: *p-antitone-iff*)
hence $f^{TT} \leq$ − $?e^T$
by (*metis conv-complement conv-dist-inf inf.orderE inf.orderI*)
hence $f \leq$ − $?e^T$
by *auto*
hence $f = f \sqcap$ − $?e^T$
using *inf.orderE* **by** *blast*
hence $f \sqcap$ − $?e^T \sqcap$ − *?p* ⊔ $(f \sqcap$ − $?e^T \sqcap ?p)^T \sqcup ?e = f \sqcap$ − *?p* ⊔ $(f \sqcap ?p)^T$ ⊔ *?e*
by *auto*
thus *?thesis* **by** *auto*
qed

The source of the *selected-edge* is contained in j, the vector describing those vertices still to be processed in the inner loop of Borůvka's algorithm.

lemma *et-below-j*:
assumes *vector j*
and *regular j*
and $j \neq bot$
shows *selected-edge h j g* ∗ *top* ≤ *j*
proof −
let *?e* = *selected-edge h j g*
let *?c* = *choose-component* (*forest-components h*) *j*
have *?e* ∗ *top* ≤ −−(*?c* ∗ $-?c^T \sqcap g$) ∗ *top*
using *comp-isotone minarc-below* **by** *blast*
also have ... = (−−(*?c* ∗ $-?c^T$) ⊓ −−*g*) ∗ *top*
by *simp*
also have ... = (*?c* ∗ $-?c^T \sqcap$ −−*g*) ∗ *top*
using *component-is-regular regular-mult-closed* **by** *auto*
also have ... = (*?c* ⊓ $-?c^T \sqcap$ −−*g*) ∗ *top*
by (*metis component-is-vector conv-complement vector-complement-closed vector-covector*)
also have ... ≤ *?c* ∗ *top*
using *inf.cobounded1 mult-left-isotone order-trans* **by** *blast*

also have ... $\leq j * top$
by (*metis assms*(2) *comp-inf.star.circ-sup-2 comp-isotone component-in-v*)
also have ... $= j$
by (*simp add*: *assms*(1))
finally show *?thesis*
by *simp*
qed

4.4.1 Components of forests and forests modulo an equivalence

We prove a number of properties about *forest-modulo-equivalence* and *forest-components*.

lemma *fc-j-eq-j-inv*:
assumes *forest h*
and *forest-components h* $* j = j$
shows *forest-components h* $* (j \sqcap -$ *choose-component* (*forest-components h*) *j*)
$= j \sqcap -$ *choose-component* (*forest-components h*) *j*
proof −
let *?c* = *choose-component* (*forest-components h*) *j*
let *?H* = *forest-components h*
have *1*:*equivalence ?H*
by (*simp add*: *assms*(1) *forest-components-equivalence*)
have *?H* $* (j \sqcap -$ *?c*) = *?H* $* j \sqcap$ *?H* $* -$ *?c*
using *1* **by** (*metis assms*(2) *equivalence-comp-dist-inf*
inf.sup-monoid.add-commute)
hence *2*: *?H* $* (j \sqcap -$ *?c*) $= j \sqcap$ *?H* $* -$ *?c*
by (*simp add*: *assms*(2))
have *3*: $j \sqcap -$ *?c* $\leq$ *?H* $* -$ *?c*
using *1* **by** (*metis assms*(2) *dedekind-1 dual-order.trans*
equivalence-comp-dist-inf inf.cobounded2)
have *?H* $*$ *?c* $\leq$ *?c*
using *component-single* **by** *auto*
hence $?H^T * ?c \leq ?c$
using *1* **by** *simp*
hence *?H* $* -$ *?c* $\leq -$ *?c*
using *component-is-regular schroeder-3-p* **by** *force*
hence $j \sqcap$ *?H* $* -$ *?c* $\leq j \sqcap -$ *?c*
using *inf.sup-right-isotone* **by** *auto*
thus *?thesis*
using *2 3 order.antisym* **by** *simp*
qed

There is a path in the *forest-modulo-equivalence* between edges a and b if and only if there is either a path in the *forest-modulo-equivalence* from a to b or one from a to c and one from c to b.

lemma *forest-modulo-equivalence-path-split-disj*:
assumes *equivalence H*
and *arc c*
and *regular a* $\wedge$ *regular b* $\wedge$ *regular c* $\wedge$ *regular d* $\wedge$ *regular H*

shows *forest-modulo-equivalence-path a b H* $(d \sqcup c) \longleftrightarrow$ *forest-modulo-equivalence-path a b H d* $\vee$ (*forest-modulo-equivalence-path a c H d* $\wedge$ *forest-modulo-equivalence-path c b H d*)
proof −
have *1*: *forest-modulo-equivalence-path a b H* $(d \sqcup c) \longrightarrow$ *forest-modulo-equivalence-path a b H d* $\vee$ (*forest-modulo-equivalence-path a c H d* $\wedge$ *forest-modulo-equivalence-path c b H d*)
proof (*rule impI*)
assume *11*: *forest-modulo-equivalence-path a b H* $(d \sqcup c)$
hence $a^T * top \leq (H * (d \sqcup c))^\star * H * b * top$
by (*simp add*: *forest-modulo-equivalence-path-def*)
also have ... $= ((H * d)^\star \sqcup (H * d)^\star * H * c * (H * d)^\star) * H * b * top$
using *assms*(*1*, *2*) *path-through-components* **by** *simp*
also have ... $= (H * d)^\star * H * b * top \sqcup (H * d)^\star * H * c * (H * d)^\star * H * b * top$
by (*simp add*: *mult-right-dist-sup*)
finally have *12*:$a^T * top \leq (H * d)^\star * H * b * top \sqcup (H * d)^\star * H * c * (H * d)^\star * H * b * top$
by *simp*
have *13*: $a^T * top \leq (H * d)^\star * H * b * top \vee a^T * top \leq (H * d)^\star * H * c * (H * d)^\star * H * b * top$
proof (*rule point-in-vector-sup*)
show *point* $(a^T * top)$
using *11 forest-modulo-equivalence-path-def mult-assoc* **by** *auto*
next
show *vector* $((H * d)^\star * H * b * top)$
using *vector-mult-closed* **by** *simp*
next
show *regular* $((H * d)^\star * H * b * top)$
using *assms*(*3*) *pp-dist-comp pp-dist-star* **by** *auto*
next
show $a^T * top \leq (H * d)^\star * H * b * top \sqcup (H * d)^\star * H * c * (H * d)^\star * H * b * top$
using *12* **by** *simp*
qed
thus *forest-modulo-equivalence-path a b H d* $\vee$ (*forest-modulo-equivalence-path a c H d* $\wedge$ *forest-modulo-equivalence-path c b H d*)
proof (*cases* $a^T * top \leq (H * d)^\star * H * b * top$)
case *True*
assume $a^T * top \leq (H * d)^\star * H * b * top$
hence *forest-modulo-equivalence-path a b H d*
using *11 forest-modulo-equivalence-path-def* **by** *auto*
thus *?thesis*
by *simp*
next
case *False*
have *14*: $a^T * top \leq (H * d)^\star * H * c * (H * d)^\star * H * b * top$
using *13* **by** (*simp add*: *False*)
hence *15*: $a^T * top \leq (H * d)^\star * H * c * top$

by (*metis mult-right-isotone order-lesseq-imp top-greatest mult-assoc*)
have $c^T * top \leq (H * d)^\star * H * b * top$
proof (*rule ccontr*)
assume $\neg\ c^T * top \leq (H * d)^\star * H * b * top$
hence $c^T * top \leq -((H * d)^\star * H * b * top)$
by (*meson assms*(*2*, *3*) *point-in-vector-or-complement regular-closed-star regular-closed-top regular-mult-closed vector-mult-closed vector-top-closed*)
hence $c * (H * d)^\star * H * b * top \leq bot$
using *schroeder-3-p mult-assoc* **by** *auto*
thus *False*
using *13 False sup.absorb-iff1 mult-assoc* **by** *auto*
qed
hence *forest-modulo-equivalence-path a c H d* $\wedge$ *forest-modulo-equivalence-path c b H d*
using *11 15 assms*(*2*) *forest-modulo-equivalence-path-def* **by** *auto*
thus *?thesis*
by *simp*
qed
qed
have *2*: *forest-modulo-equivalence-path a b H d* $\vee$ (*forest-modulo-equivalence-path a c H d* $\wedge$ *forest-modulo-equivalence-path c b H d*) $\longrightarrow$ *forest-modulo-equivalence-path a b H* $(d \sqcup c)$
proof $-$
have *21*: *forest-modulo-equivalence-path a b H d* $\longrightarrow$ *forest-modulo-equivalence-path a b H* $(d \sqcup c)$
proof (*rule impI*)
assume *22*:*forest-modulo-equivalence-path a b H d*
hence $a^T * top \leq (H * d)^\star * H * b * top$
using *forest-modulo-equivalence-path-def* **by** *blast*
hence $a^T * top \leq (H * (d \sqcup c))^\star * H * b * top$
by (*simp add*: *mult-left-isotone mult-right-dist-sup mult-right-isotone order.trans star-isotone star-slide*)
thus *forest-modulo-equivalence-path a b H* $(d \sqcup c)$
using *22 forest-modulo-equivalence-path-def* **by** *blast*
qed
have *forest-modulo-equivalence-path a c H d* $\wedge$ *forest-modulo-equivalence-path c b H d* $\longrightarrow$ *forest-modulo-equivalence-path a b H* $(d \sqcup c)$
proof (*rule impI*)
assume *23*: *forest-modulo-equivalence-path a c H d* $\wedge$ *forest-modulo-equivalence-path c b H d*
hence $a^T * top \leq (H * d)^\star * H * c * top$
using *forest-modulo-equivalence-path-def* **by** *blast*
also have ... $\leq (H * d)^\star * H * c * c^T * c * top$
by (*metis ex231c comp-inf.star.circ-sup-2 mult-isotone mult-right-isotone mult-assoc*)
also have ... $\leq (H * d)^\star * H * c * c^T * top$
by (*simp add*: *mult-right-isotone mult-assoc*)
also have ... $\leq (H * d)^\star * H * c * (H * d)^\star * H * b * top$
using *23 mult-right-isotone mult-assoc* **by** (*simp add*:

forest-modulo-equivalence-path-def)
    also have ... $\leq (H * d)^\star * H * b * top \sqcup (H * d)^\star * H * c * (H * d)^\star * H * b * top$
      by (*simp add*: *forest-modulo-equivalence-path-def*)
    finally have $a^T * top \leq (H * (d \sqcup c))^\star * H * b * top$
      using *assms*(*1*, *2*) *path-through-components mult-right-dist-sup* **by** *simp*
    thus *forest-modulo-equivalence-path a b H* $(d \sqcup c)$
      using *23 forest-modulo-equivalence-path-def* **by** *blast*
  qed
  thus *?thesis*
    using *21* **by** *auto*
qed
thus *?thesis*
  using *1 2* **by** *blast*
qed

lemma *dT-He-eq-bot*:
assumes *vector j*
  and *regular j*
  and $d * top \leq - j$
  and *forest-components h* $* j = j$
  and $j \neq bot$
shows $d^T *$ *forest-components h* $*$ *selected-edge h j g* $\leq bot$
proof −
let *?e* $=$ *selected-edge h j g*
let *?H* $=$ *forest-components h*
have *1*: *?e* $* top \leq j$
  using *assms*(*1*, *2*, *5*) *et-below-j* **by** *auto*
have $d^T * ?H * ?e \leq (d * top)^T * ?H * (?e * top)$
  by (*simp add*: *comp-isotone conv-isotone top-right-mult-increasing*)
also have ... $\leq (d * top)^T * ?H * j$
  using *1 mult-right-isotone* **by** *auto*
also have ... $\leq (- j)^T * ?H * j$
  by (*simp add*: *assms*(*3*) *conv-isotone mult-left-isotone*)
also have ... $= (- j)^T * j$
  using *assms*(*4*) *comp-associative* **by** *auto*
also have ... $=$ *bot*
  by (*simp add*: *assms*(*1*) *conv-complement covector-vector-comp*)
finally show *?thesis*
  using *coreflexive-bot-closed le-bot* **by** *blast*
qed

lemma *forest-modulo-equivalence-d-U-e*:
assumes *forest f*
  and *vector j*
  and *regular j*
  and *forest h*
  and *forest-modulo-equivalence* (*forest-components h*) *d*
  and $d * top \leq - j$

```
    and forest-components h * j = j
    and f ⊔ f^T = h ⊔ h^T ⊔ d ⊔ d^T
    and selected-edge h j g ≤ - forest-components f
    and j ≠ bot
  shows forest-modulo-equivalence (forest-components h) (d ⊔ selected-edge h j g)
proof (cases selected-edge h j g = bot)
  let ?e = selected-edge h j g
  case True
  assume ?e = bot
  thus ?thesis
    by (simp add: True assms(5))
next
  let ?H = forest-components h
  let ?F = forest-components f
  let ?e = selected-edge h j g
  let ?d' = d ⊔ ?e
  case False
  assume e-not-bot: ?e ≠ bot
  have forest-modulo-equivalence (forest-components h) (d ⊔ selected-edge h j g)
  proof (unfold forest-modulo-equivalence-def, intro conjI)
    show 01: reflexive ?H
      by (simp add: assms(4) forest-components-equivalence)
    show 02: transitive ?H
      by (simp add: assms(4) forest-components-equivalence)
    show 03: symmetric ?H
      by (simp add: assms(4) forest-components-equivalence)
    have 04: equivalence ?H
      by (simp add: 01 02 03)
    show univalent (?H * ?d')
    proof -
      have (?H * ?d')^T * (?H * ?d') = ?d'^T * ?H^T * ?H * ?d'
        using conv-dist-comp mult-assoc by auto
      also have ... = ?d'^T * ?H * ?H * ?d'
        by (simp add: conv-dist-comp conv-star-commute)
      also have ... = ?d'^T * ?H * ?d'
        using 01 02 by (metis preorder-idempotent mult-assoc)
      finally have 21: univalent (?H * ?d') ⟷ ?e^T * ?H * d ⊔ d^T * ?H * ?e ⊔
?e^T * ?H * ?e ⊔ d^T * ?H * d ≤ 1
        using conv-dist-sup semiring.distrib-left semiring.distrib-right by auto
      have 22: ?e^T * ?H * ?e ≤ 1
      proof -
        have 221: ?e^T * ?H * ?e ≤ ?e^T * top * ?e
          by (simp add: mult-left-isotone mult-right-isotone)
        have arc ?e
          using e-not-bot minarc-arc minarc-bot-iff by blast
        hence ?e^T * top * ?e ≤ 1
          using arc-expanded by blast
        thus ?thesis
          using 221 dual-order.trans by blast
```

qed
have *24*: $d^T * ?H * ?e \leq 1$
by (*metis assms*(*2*, *3*, *6*, *7*, *10*) *dT-He-eq-bot coreflexive-bot-closed le-bot*)
hence $(d^T * ?H * ?e)^T \leq 1^T$
using *conv-isotone* **by** *blast*
hence $?e^T * ?H^T * d^{TT} \leq 1$
by (*simp add*: *conv-dist-comp mult-assoc*)
hence *25*: $?e^T * ?H * d \leq 1$
using *assms*(*4*) *fch-equivalence* **by** *auto*
have *8*: $d^T * ?H * d \leq 1$
using *04 assms*(*5*) *dTransHd-le-1 forest-modulo-equivalence-def* **by** *blast*
thus *?thesis*
using *21 22 24 25* **by** *simp*
qed
show *coreflexive* $(?H \sqcap ?d' * ?d'^T)$
proof −
have *coreflexive* $(?H \sqcap ?d' * ?d'^T) \longleftrightarrow ?H \sqcap (d \sqcup ?e) * (d^T \sqcup ?e^T) \leq 1$
by (*simp add*: *conv-dist-sup*)
also have ... $\longleftrightarrow ?H \sqcap (d * d^T \sqcup d * ?e^T \sqcup ?e * d^T \sqcup ?e * ?e^T) \leq 1$
by (*metis mult-left-dist-sup mult-right-dist-sup sup.left-commute sup-commute*)
finally have *1*: *coreflexive* $(?H \sqcap ?d' * ?d'^T) \longleftrightarrow ?H \sqcap d * d^T \sqcup ?H \sqcap d * ?e^T \sqcup ?H \sqcap ?e * d^T \sqcup ?H \sqcap ?e * ?e^T \leq 1$
by (*simp add*: *inf-sup-distrib1*)
have *31*: $?H \sqcap d * d^T \leq 1$
using *assms*(*5*) *forest-modulo-equivalence-def* **by** *blast*
have *32*: $?H \sqcap ?e * d^T \leq 1$
proof −
have $?e * d^T \leq ?e * top * (d * top)^T$
by (*simp add*: *conv-isotone mult-isotone top-right-mult-increasing*)
also have ... $\leq ?e * top * - j^T$
by (*metis assms*(*6*) *conv-complement conv-isotone mult-right-isotone*)
also have ... $\leq j * - j^T$
using *assms*(*2*, *3*, *10*) *et-below-j mult-left-isotone* **by** *auto*
also have ... $\leq - ?H$
using *03* **by** (*metis assms*(*2*, *3*, *7*) *conv-complement conv-dist-comp equivalence-top-closed mult-left-isotone schroeder-3-p vector-top-closed*)
finally have $?e * d^T \leq - ?H$
by *simp*
thus *?thesis*
by (*metis inf.coboundedI1 p-antitone-iff p-shunting-swap regular-one-closed*)
qed
have *33*: $?H \sqcap d * ?e^T \leq 1$
proof −
have *331*: *injective h*
by (*simp add*: *assms*(*4*))
have $(?H \sqcap ?e * d^T)^T \leq 1$
using *32 coreflexive-conv-closed* **by** *auto*

hence *?H* ⊓ (*?e* ∗ *d*^{T})^T ≤ *1*
using *331 conv-dist-inf forest-components-equivalence* **by** *auto*
thus *?thesis*
using *conv-dist-comp* **by** *auto*
qed
have *34*: *?H* ⊓ *?e* ∗ *?e*^{T} ≤ *1*
proof −
have *341*:*arc ?e* ∧ *arc* (*?e*^{T})
using *e-not-bot minarc-arc minarc-bot-iff* **by** *auto*
have *?H* ⊓ *?e* ∗ *?e*^{T} ≤ *?e* ∗ *?e*^{T}
by *auto*
thus *?thesis*
using *341 arc-injective le-infI2* **by** *blast*
qed
thus *?thesis*
using *1 31 32 33 34* **by** *simp*
qed
show *4*:(*?H* ∗ (*d* ⊔ *?e*))$^+$ ⊓ *?H* ≤ *bot*
proof −
have *40*: (*?H* ∗ *d*)$^+$ ≤ −*?H*
using *assms*(*5*) *forest-modulo-equivalence-def bot-unique pseudo-complement* **by** *blast*
have *?e* ≤ − *?F*
by (*simp add*: *assms*(*9*))
hence *?F* ≤ − *?e*
by (*simp add*: *p-antitone-iff*)
hence *?F*^{T} ∗ *?F* ≤ − *?e*
using *assms*(*1*) *fch-equivalence* **by** *fastforce*
hence *?F*^{T} ∗ *?F* ∗ *?F*^{T} ≤ − *?e*
by (*metis assms*(*1*) *fch-equivalence forest-components-star star.circ-decompose-9*)
hence *41*: *?F* ∗ *?e* ∗ *?F* ≤ − *?F*
using *triple-schroeder-p* **by** *blast*
hence *42*:(*?F* ∗ *?F*)* ∗ *?F* ∗ *?e* ∗ (*?F* ∗ *?F*)* ≤ − *?F*
proof −
have *43*: *?F* ∗ *?F* = *?F*
using *assms*(*1*) *forest-components-equivalence preorder-idempotent* **by** *auto*
hence *?F* ∗ *?e* ∗ *?F* = *?F* ∗ *?F* ∗ *?e* ∗ *?F*
by *simp*
also have ... = (*?F*)* ∗ *?F* ∗ *?e* ∗ (*?F*)*
by (*simp add*: *assms*(*1*) *forest-components-star*)
also have ... = (*?F* ∗ *?F*)* ∗ *?F* ∗ *?e* ∗ (*?F* ∗ *?F*)*
using *43* **by** *simp*
finally show *?thesis*
using *41* **by** *simp*
qed
hence *44*: (*?H* ∗ *d*)* ∗ *?H* ∗ *?e* ∗ (*?H* ∗ *d*)* ≤ − *?H*
proof −

have *45*: *?H* ≤ *?F*
using *assms*(*1*, *4*, *8*) *H-below-F* **by** *blast*
hence *46*:*?H* ∗ *?e* ≤ *?F* ∗ *?e*
by (*simp add*: *mult-left-isotone*)
have *d* ≤ *f* ⊔ *f*^{T}
using *assms*(*8*) *sup.left-commute sup-commute* **by** *auto*
also have ... ≤ *?F*
by (*metis forest-components-increasing le-supI2 star.circ-back-loop-fixpoint star.circ-increasing sup.bounded-iff*)
finally have *d* ≤ *?F*
by *simp*
hence *?H* ∗ *d* ≤ *?F* ∗ *?F*
using *45 mult-isotone* **by** *auto*
hence *47*: (*?H* ∗ *d*)* ≤ (*?F* ∗ *?F*)*
by (*simp add*: *star-isotone*)
hence (*?H* ∗ *d*)* ∗ *?H* ∗ *?e* ∗ (*?H* ∗ *d*)* ≤ (*?H* ∗ *d*)* ∗ *?F* ∗ *?e* ∗ (*?H* ∗ *d*)*
using *46* **by** (*metis mult-left-isotone mult-right-isotone mult-assoc*)
also have ... ≤ (*?F* ∗ *?F*)* ∗ *?F* ∗ *?e* ∗ (*?F* ∗ *?F*)*
using *47 mult-left-isotone mult-right-isotone* **by** (*simp add*: *comp-isotone*)
also have ... ≤ − *?F*
using *42* **by** *simp*
also have ... ≤ − *?H*
using *45* **by** (*simp add*: *p-antitone*)
finally show *?thesis*
by *simp*
qed
have (*?H* ∗ (*d* ⊔ *?e*))$^+$ = (*?H* ∗ (*d* ⊔ *?e*))* ∗ (*?H* ∗ (*d* ⊔ *?e*))
using *star.circ-plus-same* **by** *auto*
also have ... = ((*?H* ∗ *d*)* ⊔ (*?H* ∗ *d*)* ∗ *?H* ∗ *?e* ∗ (*?H* ∗ *d*)*) ∗ (*?H* ∗ (*d* ⊔ *?e*))
using *assms*(*4*) *e-not-bot forest-components-equivalence minarc-arc minarc-bot-iff path-through-components* **by** *auto*
also have ... = (*?H* ∗ *d*)* ∗ (*?H* ∗ (*d* ⊔ *?e*)) ⊔ (*?H* ∗ *d*)* ∗ *?H* ∗ *?e* ∗ (*?H* ∗ *d*)* ∗ (*?H* ∗ (*d* ⊔ *?e*))
using *mult-right-dist-sup* **by** *auto*
also have ... = (*?H* ∗ *d*)* ∗ (*?H* ∗ *d* ⊔ *?H* ∗ *?e*) ⊔ (*?H* ∗ *d*)* ∗ *?H* ∗ *?e* ∗ (*?H* ∗ *d*)* ∗ (*?H* ∗ *d* ⊔ *?H* ∗ *?e*)
by (*simp add*: *mult-left-dist-sup*)
also have ... = (*?H* ∗ *d*)* ∗ *?H* ∗ *d* ⊔ (*?H* ∗ *d*)* ∗ *?H* ∗ *?e* ⊔ (*?H* ∗ *d*)* ∗ *?H* ∗ *?e* ∗ (*?H* ∗ *d*)* ∗ (*?H* ∗ *d* ⊔ *?H* ∗ *?e*)
using *mult-left-dist-sup mult-assoc* **by** *auto*
also have ... = (*?H* ∗ *d*)$^+$ ⊔ (*?H* ∗ *d*)* ∗ *?H* ∗ *?e* ⊔ (*?H* ∗ *d*)* ∗ *?H* ∗ *?e* ∗ (*?H* ∗ *d*)* ∗ (*?H* ∗ *d* ⊔ *?H* ∗ *?e*)
by (*simp add*: *star.circ-plus-same mult-assoc*)
also have ... = (*?H* ∗ *d*)$^+$ ⊔ (*?H* ∗ *d*)* ∗ *?H* ∗ *?e* ⊔ (*?H* ∗ *d*)* ∗ *?H* ∗ *?e* ∗ (*?H* ∗ *d*)* ∗ *?H* ∗ *d* ⊔ (*?H* ∗ *d*)* ∗ *?H* ∗ *?e* ∗ (*?H* ∗ *d*)* ∗ *?H* ∗ *?e*
by (*simp add*: *mult.semigroup-axioms semiring.distrib-left sup.semigroup-axioms semigroup.assoc*)

```
    also have ... ≤ (?H * d)^+ ⊔ (?H * d)^⋆ * ?H * ?e ⊔ (?H * d)^⋆ * ?H * ?e *
(?H * d)^⋆ * ?H * d ⊔ (?H * d)^⋆ * ?H * ?e
    proof −
      have ?e * (?H * d)^⋆ * ?H * ?e ≤ ?e * top * ?e
        by (metis comp-associative comp-inf.coreflexive-idempotent
comp-inf.coreflexive-transitive comp-isotone top.extremum)
      also have ... ≤ ?e
        using e-not-bot arc-top-arc minarc-arc minarc-bot-iff by auto
      finally have ?e * (?H * d)^⋆ * ?H * ?e ≤ ?e
        by simp
      hence (?H * d)^⋆ * ?H * ?e * (?H * d)^⋆ * ?H * ?e ≤ (?H * d)^⋆ * ?H * ?e
        by (simp add: comp-associative comp-isotone)
      thus ?thesis
        using sup-right-isotone by blast
    qed
    also have ... = (?H * d)^+ ⊔ (?H * d)^⋆ * ?H * ?e ⊔ (?H * d)^⋆ * ?H * ?e *
(?H * d)^⋆ * ?H * d
      by (simp add: order.eq-iff ac-simps)
    also have ... = (?H * d)^+ ⊔ (?H * d)^⋆ * ?H * ?e ⊔ (?H * d)^⋆ * ?H * ?e *
(?H * d)^+
      using star.circ-plus-same mult-assoc by auto
    also have ... = (?H * d)^+ ⊔ (?H * d)^⋆ * ?H * ?e * (1 ⊔ (?H * d)^+)
      by (simp add: mult-left-dist-sup sup-assoc)
    also have ... = (?H * d)^+ ⊔ (?H * d)^⋆ * ?H * ?e * (?H * d)^⋆
      by (simp add: star-left-unfold-equal)
    also have ... ≤ − ?H
      using 40 44 assms(5) sup.boundedI by blast
    finally show ?thesis
      using pseudo-complement by force
  qed
 qed
 thus ?thesis
  by blast
qed
```

4.4.2 Identifying arcs

The expression $d \sqcap \top e^\top H \sqcap (Hd^\top)^* Ha^\top \top$ identifies the edge incoming to the component that the *selected-edge*, e, is outgoing from and which is on the path from edge a to e. Here, we prove this expression is an *arc*.

```
lemma shows-arc-x:
 assumes forest-modulo-equivalence H d
   and forest-modulo-equivalence-path a e H d
   and H * d * (H * d)^⋆ ≤ − H
   and ¬ a^T * top ≤ H * e * top
   and regular a
   and regular e
   and regular H
   and regular d
```

shows *arc* $(d \sqcap top * e^T * H \sqcap (H * d^T)^\star * H * a^T * top)$
proof −
let *?x* $= d \sqcap top * e^T * H \sqcap (H * d^T)^\star * H * a^T * top$
have *1*:*regular ?x*
using *assms*(*5*, *6*, *7*, *8*) *regular-closed-star regular-conv-closed regular-mult-closed* **by** *auto*
have *2*: $a^T * top * a \leq 1$
using *arc-expanded assms*(*2*) *forest-modulo-equivalence-path-def* **by** *auto*
have *3*: $e * top * e^T \leq 1$
using *assms*(*2*) *forest-modulo-equivalence-path-def arc-expanded* **by** *blast*
have *4*: $top * ?x * top = top$
proof −
have $a^T * top \leq (H * d)^\star * H * e * top$
using *assms*(*2*) *forest-modulo-equivalence-path-def* **by** *blast*
also have ... $= H * e * top \sqcup (H * d)^\star * H * d * H * e * top$
by (*metis star.circ-loop-fixpoint star.circ-plus-same sup-commute mult-assoc*)
finally have $a^T * top \leq H * e * top \sqcup (H * d)^\star * H * d * H * e * top$
by *simp*
hence $a^T * top \leq H * e * top \vee a^T * top \leq (H * d)^\star * H * d * H * e * top$
using *assms*(*2*, *6*, *7*) *point-in-vector-sup forest-modulo-equivalence-path-def regular-mult-closed vector-mult-closed* **by** *auto*
hence $a^T * top \leq (H * d)^\star * H * d * H * e * top$
using *assms*(*4*) **by** *blast*
also have ... $= (H * d)^\star * H * d * (H * e * top \sqcap H * e * top)$
by (*simp add*: *mult-assoc*)
also have ... $= (H * d)^\star * H * (d \sqcap (H * e * top)^T) * H * e * top$
by (*metis comp-associative covector-inf-comp-3 star.circ-left-top star.circ-top*)
also have ... $= (H * d)^\star * H * (d \sqcap top^T * e^T * H^T) * H * e * top$
using *conv-dist-comp mult-assoc* **by** *auto*
also have ... $= (H * d)^\star * H * (d \sqcap top * e^T * H) * H * e * top$
using *assms*(*1*) **by** (*simp add*: *forest-modulo-equivalence-def*)
finally have *2*: $a^T * top \leq (H * d)^\star * H * (d \sqcap top * e^T * H) * H * e * top$
by *simp*
hence $e * top \leq ((H * d)^\star * H * (d \sqcap top * e^T * H) * H)^T * a^T * top$
proof −
have *bijective* $(e * top) \wedge$ *bijective* $(a^T * top)$
using *assms*(*2*) *forest-modulo-equivalence-path-def* **by** *auto*
thus *?thesis*
using *2* **by** (*metis bijective-reverse mult-assoc*)
qed
also have ... $= H^T * (d \sqcap top * e^T * H)^T * H^T * (H * d)^{\star T} * a^T * top$
by (*simp add*: *conv-dist-comp mult-assoc*)
also have ... $= H * (d \sqcap top * e^T * H)^T * H * (H * d)^{\star T} * a^T * top$
using *assms*(*1*) *forest-modulo-equivalence-def* **by** *auto*
also have ... $= H * (d \sqcap top * e^T * H)^T * H * (d^T * H)^\star * a^T * top$
using *assms*(*1*) *forest-modulo-equivalence-def conv-dist-comp conv-star-commute* **by** *auto*
also have ... $= H * (d^T \sqcap H * e * top) * H * (d^T * H)^\star * a^T * top$

using *assms(1) conv-dist-comp forest-modulo-equivalence-def comp-associative conv-dist-inf* **by** *auto*
also have ... = $H * (d^T \sqcap H * e * top) * (H * d^T)^\star * H * a^T * top$
by (*simp add*: *comp-associative star-slide*)
also have ... = $H * (d^T \sqcap H * e * top) * ((H * d^T)^\star * H * a^T * top \sqcap (H * d^T)^\star * H * a^T * top)$
using *mult-assoc* **by** *auto*
also have ... = $H * (d^T \sqcap H * e * top \sqcap ((H * d^T)^\star * H * a^T * top)^T) * (H * d^T)^\star * H * a^T * top$
by (*smt comp-inf-vector covector-comp-inf vector-conv-covector vector-top-closed mult-assoc*)
also have ... = $H * (d^T \sqcap (top * e^T * H)^T \sqcap ((H * d^T)^\star * H * a^T * top)^T) * (H * d^T)^\star * H * a^T * top$
using *assms(1) forest-modulo-equivalence-def conv-dist-comp mult-assoc* **by** *auto*
also have ... = $H * (d \sqcap top * e^T * H \sqcap (H * d^T)^\star * H * a^T * top)^T * (H * d^T)^\star * H * a^T * top$
by (*simp add*: *conv-dist-inf*)
finally have *3*: $e * top \leq H * ?x^T * (H * d^T)^\star * H * a^T * top$
by *auto*
have $?x \neq bot$
proof (*rule ccontr*)
assume $\neg\ ?x \neq bot$
hence $e * top = bot$
using *3 le-bot* **by** *auto*
thus *False*
using *assms(2, 4) forest-modulo-equivalence-path-def mult-assoc semiring.mult-zero-right* **by** *auto*
qed
thus *?thesis*
using *1* **using** *tarski* **by** *blast*
qed
have *5*: $?x * top * ?x^T \leq 1$
proof −
have *51*: $H * (d * H)^\star \sqcap d * H * d^T \leq 1$
proof −
have *511*: $d * (H * d)^\star \leq - H$
using *assms(1, 3) forest-modulo-equivalence-def preorder-idempotent schroeder-4-p triple-schroeder-p* **by** *fastforce*
hence $(d * H)^\star * d \leq - H$
using *star-slide* **by** *auto*
hence $H * (d^T * H)^\star \leq - d$
by (*smt assms(1) forest-modulo-equivalence-def conv-dist-comp conv-star-commute schroeder-4-p star-slide*)
hence $H * (d * H)^\star \leq - d^T$
using *511* **by** (*metis assms(1) forest-modulo-equivalence-def schroeder-5-p star-slide*)
hence $H * (d * H)^\star \leq - (H * d^T)$
by (*metis assms(3) p-antitone-iff schroeder-4-p star-slide mult-assoc*)

hence $H * (d * H)^\star \sqcap H * d^T \leq bot$
by (*simp add*: *bot-unique pseudo-complement*)
hence $H * d * (H * (d * H)^\star \sqcap H * d^T) \leq 1$
by (*simp add*: *bot-unique*)
hence *512*: $H * d * H * (d * H)^\star \sqcap H * d * H * d^T \leq 1$
using *univalent-comp-left-dist-inf assms*(*1*) *forest-modulo-equivalence-def mult-assoc* **by** *fastforce*
hence *513*: $H * d * H * (d * H)^\star \sqcap d * H * d^T \leq 1$
proof −
have $d * H * d^T \leq H * d * H * d^T$
by (*metis assms*(*1*) *forest-modulo-equivalence-def conv-dist-comp conv-involutive mult-1-right mult-left-isotone*)
thus *?thesis*
using *512* **by** (*smt dual-order.trans p-antitone p-shunting-swap regular-one-closed*)
qed
have $d^T * H * d \leq 1 \sqcup - H$
using *assms*(*1*) *forest-modulo-equivalence-def dTransHd-le-1 le-supI1* **by** *blast*
hence $(- 1 \sqcap H) * d^T * H \leq - d^T$
by (*metis assms*(*1*) *forest-modulo-equivalence-def dTransHd-le-1 inf.sup-monoid.add-commute le-infI2 p-antitone-iff regular-one-closed schroeder-4-p mult-assoc*)
hence $d * (- 1 \sqcap H) * d^T \leq - H$
by (*metis assms*(*1*) *forest-modulo-equivalence-def conv-dist-comp schroeder-3-p triple-schroeder-p*)
hence $H \sqcap d * (- 1 \sqcap H) * d^T \leq 1$
by (*metis inf.coboundedI1 p-antitone-iff p-shunting-swap regular-one-closed*)
hence $H \sqcap d * d^T \sqcup H \sqcap d * (- 1 \sqcap H) * d^T \leq 1$
using *assms*(*1*) *forest-modulo-equivalence-def le-supI* **by** *blast*
hence $H \sqcap (d * 1 * d^T \sqcup d * (- 1 \sqcap H) * d^T) \leq 1$
using *comp-inf.semiring.distrib-left* **by** *auto*
hence $H \sqcap (d * (1 \sqcup (- 1 \sqcap H)) * d^T) \leq 1$
by (*simp add*: *mult-left-dist-sup mult-right-dist-sup*)
hence *514*: $H \sqcap d * H * d^T \leq 1$
by (*metis assms*(*1*) *forest-modulo-equivalence-def comp-inf.semiring.distrib-left inf.le-iff-sup inf.sup-monoid.add-commute inf-top-right regular-one-closed stone*)
thus *?thesis*
proof −
have $H \sqcap d * H * d^T \sqcup H * d * H * (d * H)^\star \sqcap d * H * d^T \leq 1$
using *513 514* **by** *simp*
hence $d * H * d^T \sqcap (H \sqcup H * d * H * (d * H)^\star) \leq 1$
by (*simp add*: *comp-inf.semiring.distrib-left inf.sup-monoid.add-commute*)
hence $d * H * d^T \sqcap H * (1 \sqcup d * H * (d * H)^\star) \leq 1$
by (*simp add*: *mult-left-dist-sup mult-assoc*)
thus *?thesis*
by (*simp add*: *inf.sup-monoid.add-commute star-left-unfold-equal*)

qed
qed
have $?x * top * ?x^T = (d \sqcap top * e^T * H \sqcap (H * d^T)^\star * H * a^T * top) * top * (d^T \sqcap H^T * e^{TT} * top^T \sqcap top^T * a^{TT} * H^T * (d^{TT} * H^T)^\star)$
by (*simp add*: *conv-dist-comp conv-dist-inf conv-star-commute mult-assoc*)
also have $... = (d \sqcap top * e^T * H \sqcap (H * d^T)^\star * H * a^T * top) * top * (d^T \sqcap H * e * top \sqcap top * a * H * (d * H)^\star)$
using *assms*(*1*) *forest-modulo-equivalence-def* **by** *auto*
also have $... = (H * d^T)^\star * H * a^T * top \sqcap (d \sqcap top * e^T * H) * top * (d^T \sqcap H * e * top \sqcap top * a * H * (d * H)^\star)$
by (*metis inf-vector-comp vector-export-comp*)
also have $... = (H * d^T)^\star * H * a^T * top \sqcap (d \sqcap top * e^T * H) * top * top * (d^T \sqcap H * e * top \sqcap top * a * H * (d * H)^\star)$
by (*simp add*: *vector-mult-closed*)
also have $... = (H * d^T)^\star * H * a^T * top \sqcap d * ((top * e^T * H)^T \sqcap top) * top * (d^T \sqcap H * e * top \sqcap top * a * H * (d * H)^\star)$
by (*simp add*: *covector-comp-inf-1 covector-mult-closed*)
also have $... = (H * d^T)^\star * H * a^T * top \sqcap d * ((top * e^T * H)^T \sqcap (H * e * top)^T) * d^T \sqcap top * a * H * (d * H)^\star$
by (*smt comp-associative comp-inf.star-star-absorb comp-inf-vector conv-star-commute covector-comp-inf covector-conv-vector fc-top star.circ-top total-conv-surjective vector-conv-covector vector-inf-comp*)
also have $... = (H * d^T)^\star * H * a^T * top \sqcap top * a * H * (d * H)^\star \sqcap d * ((top * e^T * H)^T \sqcap (H * e * top)^T) * d^T$
using *inf.sup-monoid.add-assoc inf.sup-monoid.add-commute* **by** *auto*
also have $... = (H * d^T)^\star * H * a^T * top * top * a * H * (d * H)^\star \sqcap d * ((top * e^T * H)^T \sqcap (H * e * top)^T) * d^T$
by (*smt comp-inf.star.circ-decompose-9 comp-inf.star-star-absorb comp-inf-covector fc-top star.circ-decompose-11 star.circ-top vector-export-comp*)
also have $... = (H * d^T)^\star * H * a^T * top * a * H * (d * H)^\star \sqcap d * (H * e * top \sqcap top * e^T * H) * d^T$
using *assms*(*1*) *forest-modulo-equivalence-def conv-dist-comp mult-assoc* **by** *auto*
also have $... = (H * d^T)^\star * H * a^T * top * a * H * (d * H)^\star \sqcap d * H * e * top * e^T * H * d^T$
by (*metis comp-inf-covector inf-top.left-neutral mult-assoc*)
also have $... \leq (H * d^T)^\star * (H * d)^\star * H \sqcap d * H * e * top * e^T * H * d^T$
proof −
have $(H * d^T)^\star * H * a^T * top * a * H * (d * H)^\star \leq (H * d^T)^\star * H * 1 * H * (d * H)^\star$
using *2* **by** (*metis comp-associative comp-isotone mult-left-isotone mult-semi-associative star.circ-transitive-equal*)
also have $... = (H * d^T)^\star * H * (d * H)^\star$
using *assms*(*1*) *forest-modulo-equivalence-def mult.semigroup-axioms preorder-idempotent semigroup.assoc* **by** *fastforce*
also have $... = (H * d^T)^\star * (H * d)^\star * H$
by (*metis star-slide mult-assoc*)
finally show *?thesis*
using *inf.sup-left-isotone* **by** *auto*

qed
also have ... $\leq (H * d^T)^\star * (H * d)^\star * H \sqcap d * H * d^T$
proof −
have $d * H * e * top * e^T * H * d^T \leq d * H * 1 * H * d^T$
using *3* **by** (*metis comp-isotone idempotent-one-closed mult-left-isotone mult-sub-right-one mult-assoc*)
also have ... $\leq d * H * d^T$
by (*metis assms(1) forest-modulo-equivalence-def mult-left-isotone mult-one-associative mult-semi-associative preorder-idempotent*)
finally show *?thesis*
using *inf.sup-right-isotone* **by** *auto*
qed
also have ... $= H * (d^T * H)^\star * (H * d)^\star * H \sqcap d * H * d^T$
by (*metis assms(1) forest-modulo-equivalence-def comp-associative preorder-idempotent star-slide*)
also have ... $= H * ((d^T * H)^\star \sqcup (H * d)^\star) * H \sqcap d * H * d^T$
by (*simp add*: *assms(1) expand-forest-modulo-equivalence mult.semigroup-axioms semigroup.assoc*)
also have ... $= (H * (d^T * H)^\star * H \sqcup H * (H * d)^\star * H) \sqcap d * H * d^T$
by (*simp add*: *mult-left-dist-sup mult-right-dist-sup*)
also have ... $= (H * d^T)^\star * H \sqcap d * H * d^T \sqcup H * (d * H)^\star \sqcap d * H * d^T$
by (*smt assms(1) forest-modulo-equivalence-def inf-sup-distrib2 mult.semigroup-axioms preorder-idempotent star-slide semigroup.assoc*)
also have ... $\leq (H * d^T)^\star * H \sqcap d * H * d^T \sqcup 1$
using *51 comp-inf.semiring.add-left-mono* **by** *blast*
finally have *?x* $* top * ?x^T \leq 1$
using *51* **by** (*smt assms(1) forest-modulo-equivalence-def conv-dist-comp conv-dist-inf conv-dist-sup conv-involutive conv-star-commute equivalence-one-closed mult.semigroup-axioms sup.absorb2 semigroup.assoc conv-isotone conv-order*)
thus *?thesis*
by *simp*
qed
have *6*: *?x*$^T * top * ?x \leq 1$
proof −
have *?x*$^T * top * ?x = (d^T \sqcap H^T * e^{TT} * top^T \sqcap top^T * a^{TT} * H^T * (d^{TT} * H^T)^\star) * top * (d \sqcap top * e^T * H \sqcap (H * d^T)^\star * H * a^T * top)$
by (*simp add*: *conv-dist-comp conv-dist-inf conv-star-commute mult-assoc*)
also have ... $= (d^T \sqcap H * e * top \sqcap top * a * H * (d * H)^\star) * top * (d \sqcap top * e^T * H \sqcap (H * d^T)^\star * H * a^T * top)$
using *assms(1) forest-modulo-equivalence-def* **by** *auto*
also have ... $= H * e * top \sqcap (d^T \sqcap top * a * H * (d * H)^\star) * top * (d \sqcap top * e^T * H \sqcap (H * d^T)^\star * H * a^T * top)$
by (*smt comp-associative inf.sup-monoid.add-assoc inf.sup-monoid.add-commute star.circ-left-top star.circ-top vector-inf-comp*)
also have ... $= H * e * top \sqcap d^T * ((top * a * H * (d * H)^\star)^T \sqcap top) * (d \sqcap top * e^T * H \sqcap (H * d^T)^\star * H * a^T * top)$
by (*simp add*: *covector-comp-inf-1 covector-mult-closed*)
also have ... $= H * e * top \sqcap d^T * (d * H)^{\star T} * H * a^T * top * (d \sqcap top *$

$e^T * H \sqcap (H * d^T)^\star * H * a^T * top$)
using *assms(1) forest-modulo-equivalence-def comp-associative conv-dist-comp* **by** *auto*
also have ... $= H * e * top \sqcap d^T * (d * H)^{\star T} * H * a^T * top * (d \sqcap (H * d^T)^\star * H * a^T * top) \sqcap top * e^T * H$
by (*smt comp-associative comp-inf-covector inf.sup-monoid.add-assoc inf.sup-monoid.add-commute*)
also have ... $= H * e * top \sqcap d^T * (d * H)^{\star T} * H * a^T * (top \sqcap ((H * d^T)^\star * H * a^T * top)^T) * d \sqcap top * e^T * H$
by (*metis comp-associative comp-inf-vector vector-conv-covector vector-top-closed*)
also have ... $= H * e * top \sqcap (H * e * top)^T \sqcap d^T * (d * H)^{\star T} * H * a^T * ((H * d^T)^\star * H * a^T * top)^T * d$
by (*smt assms(1) forest-modulo-equivalence-def conv-dist-comp inf.left-commute inf.sup-monoid.add-commute symmetric-top-closed mult-assoc inf-top.left-neutral*)
also have ... $= H * e * top * (H * e * top)^T \sqcap d^T * (d * H)^{\star T} * H * a^T * ((H * d^T)^\star * H * a^T * top)^T * d$
using *vector-covector vector-mult-closed* **by** *auto*
also have ... $= H * e * top * top^T * e^T * H^T \sqcap d^T * (d * H)^{\star T} * H * a^T * top^T * a^{TT} * H^T * (H * d^T)^{\star T} * d$
by (*smt conv-dist-comp mult.semigroup-axioms symmetric-top-closed semigroup.assoc*)
also have ... $= H * e * top * top * e^T * H \sqcap d^T * (H * d^T)^\star * H * a^T * top * a * H * (d * H)^\star * d$
using *assms(1) forest-modulo-equivalence-def conv-dist-comp conv-star-commute* **by** *auto*
also have ... $= H * e * top * e^T * H \sqcap d^T * (H * d^T)^\star * H * a^T * top * a * H * (d * H)^\star * d$
using *vector-top-closed mult-assoc* **by** *auto*
also have ... $\leq H \sqcap d^T * (H * d^T)^\star * H * (d * H)^\star * d$
proof −
have $H * e * top * e^T * H \leq H * 1 * H$
using *3* **by** (*metis comp-associative mult-left-isotone mult-right-isotone*)
also have ... $= H$
using *assms(1) forest-modulo-equivalence-def preorder-idempotent* **by** *auto*
finally have *611*: $H * e * top * e^T * H \leq H$
by *simp*
have $d^T * (H * d^T)^\star * H * a^T * top * a * H * (d * H)^\star * d \leq d^T * (H * d^T)^\star * H * 1 * H * (d * H)^\star * d$
using *2* **by** (*metis comp-associative mult-left-isotone mult-right-isotone*)
also have ... $= d^T * (H * d^T)^\star * H * (d * H)^\star * d$
using *assms(1) forest-modulo-equivalence-def mult.semigroup-axioms preorder-idempotent semigroup.assoc* **by** *fastforce*
finally have $d^T * (H * d^T)^\star * H * a^T * top * a * H * (d * H)^\star * d \leq d^T * (H * d^T)^\star * H * (d * H)^\star * d$
by *simp*
thus *?thesis*
using *611 comp-inf.comp-isotone* **by** *blast*

```
    qed
    also have ... = H ⊓ (d^T * H)^⋆ * d^T * H * d * (H * d)^⋆
      using star-slide mult-assoc by auto
    also have ... ≤ H ⊓ (d^T * H)^⋆ * (H * d)^⋆
    proof −
      have (d^T * H)^⋆ * d^T * H * d * (H * d)^⋆ ≤ (d^T * H)^⋆ * 1 * (H * d)^⋆
        by (smt assms(1) forest-modulo-equivalence-def conv-dist-comp
mult-left-isotone mult-right-isotone preorder-idempotent mult-assoc)
      also have ... = (d^T * H)^⋆ * (H * d)^⋆
        by simp
      finally show ?thesis
        using inf.sup-right-isotone by blast
    qed
    also have ... = H ⊓ ((d^T * H)^⋆ ⊔ (H * d)^⋆)
      by (simp add: assms(1) expand-forest-modulo-equivalence)
    also have ... = H ⊓ (d^T * H)^⋆ ⊔ H ⊓ (H * d)^⋆
      by (simp add: comp-inf.semiring.distrib-left)
    also have ... = 1 ⊔ H ⊓ (d^T * H)^+ ⊔ H ⊓ (H * d)^+
    proof −
      have 612: H ⊓ (H * d)^⋆ = 1 ⊔ H ⊓ (H * d)^+
        using assms(1) forest-modulo-equivalence-def reflexive-inf-star by blast
      have H ⊓ (d^T * H)^⋆ = 1 ⊔ H ⊓ (d^T * H)^+
        using assms(1) forest-modulo-equivalence-def reflexive-inf-star by auto
      thus ?thesis
        using 612 sup-assoc sup-commute by auto
    qed
    also have ... ≤ 1
    proof −
      have 613: H ⊓ (H * d)^+ ≤ 1
        by (metis assms(3) inf.coboundedI1 p-antitone-iff p-shunting-swap
regular-one-closed)
      hence H ⊓ (d^T * H)^+ ≤ 1
        by (metis assms(1) forest-modulo-equivalence-def conv-dist-comp
conv-dist-inf conv-plus-commute coreflexive-symmetric)
      thus ?thesis
        by (simp add: 613)
    qed
    finally show ?thesis
      by simp
  qed
  have 7:bijective (?x * top)
    using 4 5 6 arc-expanded by blast
  have bijective (?x^T * top)
    using 4 5 6 arc-expanded by blast
  thus ?thesis
    using 7 by simp
qed
```

To maintain that f can be extended to a minimum spanning forest we identify an edge, $i = v \sqcap \overline{F}e\top \sqcap \top e^\top F$, that may be exchanged with the

selected-edge, *e*. Here, we show that *i* is an *arc*.

lemma *boruvka-edge-arc*:
assumes *equivalence F*
and *forest v*
and *arc e*
and *regular F*
and $F \leq$ *forest-components* $(F \sqcap v)$
and *regular v*
and $v * e^T = bot$
and $e * F * e = bot$
and $e^T \leq v^\star$
and $e \neq bot$
shows *arc* $(v \sqcap -F * e * top \sqcap top * e^T * F)$
proof −
let $?i = v \sqcap -F * e * top \sqcap top * e^T * F$
have *1*: $?i^T * top * ?i \leq 1$
proof −
have $?i^T * top * ?i = (v^T \sqcap top * e^T * -F \sqcap F * e * top) * top * (v \sqcap -F * e * top \sqcap top * e^T * F)$
using *assms*(*1*) *conv-complement conv-dist-comp conv-dist-inf mult.semigroup-axioms semigroup.assoc* **by** *fastforce*
also have ... $= F * e * top \sqcap (v^T \sqcap top * e^T * -F) * top * (v \sqcap -F * e * top) \sqcap top * e^T * F$
by (*smt covector-comp-inf covector-mult-closed inf-vector-comp vector-export-comp vector-top-closed*)
also have ... $= F * e * top \sqcap (v^T \sqcap top * e^T * -F) * top * top * (v \sqcap -F * e * top) \sqcap top * e^T * F$
by (*simp add*: *comp-associative*)
also have ... $= F * e * top \sqcap v^T * (top \sqcap (top * e^T * -F)^T) * top * (v \sqcap -F * e * top) \sqcap top * e^T * F$
using *comp-associative comp-inf-vector-1* **by** *auto*
also have ... $= F * e * top \sqcap v^T * (top \sqcap (top * e^T * -F)^T) * (top \sqcap (-F * e * top)^T) * v \sqcap top * e^T * F$
by (*smt comp-inf-vector conv-dist-comp mult.semigroup-axioms symmetric-top-closed semigroup.assoc*)
also have ... $= F * e * top \sqcap v^T * (top * e^T * -F)^T * (-F * e * top)^T * v \sqcap top * e^T * F$
by *simp*
also have ... $= F * e * top \sqcap v^T * -F^T * e^{TT} * top^T * top^T * e^T * -F^T * v \sqcap top * e^T * F$
by (*metis comp-associative conv-complement conv-dist-comp*)
also have ... $= F * e * top \sqcap v^T * -F * e * top * top * e^T * -F * v \sqcap top * e^T * F$
by (*simp add*: *assms*(*1*))
also have ... $= F * e * top \sqcap v^T * -F * e * top \sqcap top * e^T * -F * v \sqcap top * e^T * F$
by (*metis comp-associative comp-inf-covector inf.sup-monoid.add-assoc inf-top.left-neutral vector-top-closed*)
also have ... $= (F \sqcap v^T * -F) * e * top \sqcap top * e^T * -F * v \sqcap top * e^T * F$

using *assms(3) injective-comp-right-dist-inf mult-assoc* **by** *auto*
also have ... = $(F \sqcap v^T * -F) * e * top \sqcap top * e^T * (F \sqcap -F * v)$
using *assms(3) conv-dist-comp inf.sup-monoid.add-assoc inf.sup-monoid.add-commute mult.semigroup-axioms univalent-comp-left-dist-inf semigroup.assoc* **by** *fastforce*
also have ... = $(F \sqcap v^T * -F) * e * top * top * e^T * (F \sqcap -F * v)$
by (*metis comp-associative comp-inf-covector inf-top.left-neutral vector-top-closed*)
also have ... = $(F \sqcap v^T * -F) * e * top * e^T * (F \sqcap -F * v)$
by (*simp add*: *comp-associative*)
also have ... ≤ $(F \sqcap v^T * -F) * (F \sqcap -F * v)$
by (*smt assms(3) conv-dist-comp mult-left-isotone shunt-bijective symmetric-top-closed top-right-mult-increasing mult-assoc*)
also have ... ≤ $(F \sqcap v^T * -F) * (F \sqcap -F * v) \sqcap F$
by (*metis assms(1) inf.absorb1 inf.cobounded1 mult-isotone preorder-idempotent*)
also have ... ≤ $(F \sqcap v^T * -F) * (F \sqcap -F * v) \sqcap (F \sqcap v)^{T\star} * (F \sqcap v)^\star$
using *assms(5) comp-inf.mult-right-isotone* **by** *auto*
also have ... ≤ $(-F \sqcap v^T) * -F * -F * (-F \sqcap v) \sqcap (F \sqcap v)^{T\star} * (F \sqcap v)^\star$
proof −
have $F \sqcap v^T * -F \leq (v^T \sqcap F * -F^T) * -F$
by (*metis conv-complement dedekind-2 inf-commute*)
also have ... = $(v^T \sqcap -F^T) * -F$
using *assms(1) equivalence-comp-left-complement* **by** *simp*
finally have $F \sqcap v^T * -F \leq F \sqcap (v^T \sqcap -F) * -F$
using *assms(1)* **by** *auto*
hence *11*: $F \sqcap v^T * -F = F \sqcap (-F \sqcap v^T) * -F$
by (*metis inf.antisym-conv inf.sup-monoid.add-commute comp-left-subdist-inf inf.boundedE inf.sup-right-isotone*)
hence $F^T \sqcap -F^T * v^{TT} = F^T \sqcap -F^T * (-F^T \sqcap v^{TT})$
by (*metis* (*full-types*) *assms(1) conv-complement conv-dist-comp conv-dist-inf*)
hence *12*: $F \sqcap -F * v = F \sqcap -F * (-F \sqcap v)$
using *assms(1)* **by** (*simp add*: *abel-semigroup.commute inf.abel-semigroup-axioms*)
have $(F \sqcap v^T * -F) * (F \sqcap -F * v) = (F \sqcap (-F \sqcap v^T) * -F) * (F \sqcap -F * (-F \sqcap v))$
using *11 12* **by** *auto*
also have ... ≤ $(-F \sqcap v^T) * -F * -F * (-F \sqcap v)$
by (*metis comp-associative comp-isotone inf.cobounded2*)
finally show *?thesis*
using *comp-inf.mult-left-isotone* **by** *blast*
qed
also have ... = $((-F \sqcap v^T) * -F * -F * (-F \sqcap v) \sqcap (F \sqcap v)^T * (F \sqcap v)^{T\star} * (F \sqcap v)^\star) \sqcup ((-F \sqcap v^T) * -F * -F * (-F \sqcap v) \sqcap (F \sqcap v)^\star)$
by (*metis comp-associative inf-sup-distrib1 star.circ-loop-fixpoint*)
also have ... = $((-F \sqcap v^T) * -F * -F * (-F \sqcap v) \sqcap (F \sqcap v^T) * (F \sqcap v)^{T\star} * (F \sqcap v)^\star) \sqcup ((-F \sqcap v^T) * -F * -F * (-F \sqcap v) \sqcap (F \sqcap v)^\star)$
using *assms(1) conv-dist-inf* **by** *auto*

also have ... = $bot \sqcup ((-F \sqcap v^T) * -F * -F * (-F \sqcap v) \sqcap (F \sqcap v)^\star)$
proof −
have $(-F \sqcap v^T) * -F * -F * (-F \sqcap v) \sqcap (F \sqcap v^T) * (F \sqcap v)^{T\star} * (F \sqcap v)^\star \leq bot$
using *assms*(*1*, *2*) *forests-bot-2* **by** (*simp add*: *comp-associative*)
thus *?thesis*
using *le-bot* **by** *blast*
qed
also have ... = $(-F \sqcap v^T) * -F * -F * (-F \sqcap v) \sqcap (1 \sqcup (F \sqcap v)^\star * (F \sqcap v))$
by (*simp add*: *star.circ-plus-same star-left-unfold-equal*)
also have ... = $((-F \sqcap v^T) * -F * -F * (-F \sqcap v) \sqcap 1) \sqcup ((-F \sqcap v^T) * -F * -F * (-F \sqcap v) \sqcap (F \sqcap v)^\star * (F \sqcap v))$
by (*simp add*: *comp-inf.semiring.distrib-left*)
also have ... ≤ $1 \sqcup ((-F \sqcap v^T) * -F * -F * (-F \sqcap v) \sqcap (F \sqcap v)^\star * (F \sqcap v))$
using *sup-left-isotone* **by** *auto*
also have ... ≤ $1 \sqcup bot$
using *assms*(*1*, *2*) *forests-bot-3 comp-inf.semiring.add-left-mono* **by** *simp*
finally show *?thesis*
by *simp*
qed
have *2*: $?i * top * ?i^T \leq 1$
proof −
have $?i * top * ?i^T = (v \sqcap -F * e * top \sqcap top * e^T * F) * top * (v^T \sqcap (-F * e * top)^T \sqcap (top * e^T * F)^T)$
by (*simp add*: *conv-dist-inf*)
also have ... = $(v \sqcap -F * e * top \sqcap top * e^T * F) * top * (v^T \sqcap top^T * e^T * -F^T \sqcap F^T * e^{TT} * top^T)$
by (*simp add*: *conv-complement conv-dist-comp mult-assoc*)
also have ... = $(v \sqcap -F * e * top \sqcap top * e^T * F) * top * (v^T \sqcap top * e^T * -F \sqcap F * e * top)$
by (*simp add*: *assms*(*1*))
also have ... = $-F * e * top \sqcap (v \sqcap top * e^T * F) * top * (v^T \sqcap top * e^T * -F \sqcap F * e * top)$
by (*smt inf.left-commute inf.sup-monoid.add-assoc vector-export-comp*)
also have ... = $-F * e * top \sqcap (v \sqcap top * e^T * F) * top * (v^T \sqcap F * e * top) \sqcap top * e^T * -F$
by (*smt comp-inf-covector inf.sup-monoid.add-assoc inf.sup-monoid.add-commute mult-assoc*)
also have ... = $-F * e * top \sqcap (v \sqcap top * e^T * F) * top * top * (v^T \sqcap F * e * top) \sqcap top * e^T * -F$
by (*simp add*: *mult-assoc*)
also have ... = $-F * e * top \sqcap v * ((top * e^T * F)^T \sqcap top) * top * (v^T \sqcap F * e * top) \sqcap top * e^T * -F$
by (*simp add*: *comp-inf-vector-1 mult.semigroup-axioms semigroup.assoc*)
also have ... = $-F * e * top \sqcap v * ((top * e^T * F)^T \sqcap top) * (top \sqcap (F * e * top)^T) * v^T \sqcap top * e^T * -F$
by (*smt comp-inf-vector covector-comp-inf vector-conv-covector*

vector-mult-closed vector-top-closed)
also have ... = $-F * e * top \sqcap v * (top * e^T * F)^T * (F * e * top)^T * v^T \sqcap top * e^T * -F$
by *simp*
also have ... = $-F * e * top \sqcap v * F^T * e^{TT} * top^T * top^T * e^T * F^T * v^T \sqcap top * e^T * -F$
by (*metis comp-associative conv-dist-comp*)
also have ... = $-F * e * top \sqcap v * F * e * top * top * e^T * F * v^T \sqcap top * e^T * -F$
using *assms*(*1*) **by** *auto*
also have ... = $-F * e * top \sqcap v * F * e * top \sqcap top * e^T * F * v^T \sqcap top * e^T * -F$
by (*smt comp-associative comp-inf-covector inf.sup-monoid.add-assoc inf-top.left-neutral vector-top-closed*)
also have ... = $(-F \sqcap v * F) * e * top \sqcap top * e^T * F * v^T \sqcap top * e^T * -F$
using *injective-comp-right-dist-inf assms*(*3*) *mult.semigroup-axioms semigroup.assoc* **by** *fastforce*
also have ... = $(-F \sqcap v * F) * e * top \sqcap top * e^T * (F * v^T \sqcap -F)$
using *injective-comp-right-dist-inf assms*(*3*) *conv-dist-comp inf.sup-monoid.add-assoc mult.semigroup-axioms univalent-comp-left-dist-inf semigroup.assoc* **by** *fastforce*
also have ... = $(-F \sqcap v * F) * e * top * top * e^T * (F * v^T \sqcap -F)$
by (*metis inf-top-right vector-export-comp vector-top-closed*)
also have ... = $(-F \sqcap v * F) * e * top * e^T * (F * v^T \sqcap -F)$
by (*simp add*: *comp-associative*)
also have ... $\leq (-F \sqcap v * F) * (F * v^T \sqcap -F)$
by (*smt assms*(*3*) *conv-dist-comp mult.semigroup-axioms mult-left-isotone shunt-bijective symmetric-top-closed top-right-mult-increasing semigroup.assoc*)
also have ... = $(-F \sqcap v * F) * ((v * F)^T \sqcap -F)$
by (*simp add*: *assms*(*1*) *conv-dist-comp*)
also have ... = $(-F \sqcap v * F) * (-F \sqcap v * F)^T$
using *assms*(*1*) *conv-complement conv-dist-inf* **by** (*simp add*: *inf.sup-monoid.add-commute*)
also have ... $\leq (-F \sqcap v) * (F \sqcap v)^\star * (F \sqcap v)^{T\star} * (-F \sqcap v)^T$
proof −
let *?Fv* = $F \sqcap v$
have $-F \sqcap v * F \leq -F \sqcap v * (F \sqcap v)^{T\star} * (F \sqcap v)^\star$
using *assms*(*5*) *inf.sup-right-isotone mult-right-isotone comp-associative* **by** *auto*
also have ... $\leq -F \sqcap v * (F \sqcap v)^\star$
proof −
have $v * v^T \leq 1$
by (*simp add*: *assms*(*2*))
hence $v * v^T * F \leq F$
using *assms*(*1*) *dual-order.trans mult-left-isotone* **by** *blast*
hence $v * v^T * F^{T\star} * F^\star \leq F$
by (*metis assms*(*1*) *mult-1-right preorder-idempotent star.circ-sup-one-right-unfold star.circ-transitive-equal star-one star-simulation-right-equal mult-assoc*)

hence $v * (F \sqcap v)^T * F^{T\star} * F^\star \leq F$
by (*meson conv-isotone dual-order.trans inf.cobounded2 inf.sup-monoid.add-commute mult-left-isotone mult-right-isotone*)
hence $v * (F \sqcap v)^T * (F \sqcap v)^{T\star} * (F \sqcap v)^\star \leq F$
by (*meson conv-isotone dual-order.trans inf.cobounded2 inf.sup-monoid.add-commute mult-left-isotone mult-right-isotone comp-isotone conv-dist-inf inf.cobounded1 star-isotone*)
hence $-F \sqcap v * (F \sqcap v)^T * (F \sqcap v)^{T\star} * (F \sqcap v)^\star \leq bot$
using *order.eq-iff p-antitone pseudo-complement* **by** *auto*
hence $(-F \sqcap v * (F \sqcap v)^T * (F \sqcap v)^{T\star} * (F \sqcap v)^\star) \sqcup v * (v \sqcap F)^\star \leq v * (v \sqcap F)^\star$
using *bot-least le-bot* **by** *fastforce*
hence $(-F \sqcup v * (v \sqcap F)^\star) \sqcap (v * (F \sqcap v)^T * (F \sqcap v)^{T\star} * (F \sqcap v)^\star \sqcup v * (v \sqcap F)^\star) \leq v * (v \sqcap F)^\star$
by (*simp add*: *sup-inf-distrib2*)
hence $(-F \sqcup v * (v \sqcap F)^\star) \sqcap v * ((F \sqcap v)^T * (F \sqcap v)^{T\star} \sqcup 1) * (v \sqcap F)^\star \leq v * (v \sqcap F)^\star$
by (*simp add*: *inf.sup-monoid.add-commute mult.semigroup-axioms mult-left-dist-sup mult-right-dist-sup semigroup.assoc*)
hence $(-F \sqcup v * (v \sqcap F)^\star) \sqcap v * (F \sqcap v)^{T\star} * (v \sqcap F)^\star \leq v * (v \sqcap F)^\star$
by (*simp add*: *star-left-unfold-equal sup-commute*)
hence $-F \sqcap v * (F \sqcap v)^{T\star} * (v \sqcap F)^\star \leq v * (v \sqcap F)^\star$
using *comp-inf.mult-right-sub-dist-sup-left inf.order-lesseq-imp* **by** *blast*
thus *?thesis*
by (*simp add*: *inf.sup-monoid.add-commute*)
qed
also have ... $\leq (v \sqcap -F * (F \sqcap v)^{T\star}) * (F \sqcap v)^\star$
by (*metis dedekind-2 conv-star-commute inf.sup-monoid.add-commute*)
also have ... $\leq (v \sqcap -F * F^{T\star}) * (F \sqcap v)^\star$
using *conv-isotone inf.sup-right-isotone mult-left-isotone mult-right-isotone star-isotone* **by** *auto*
also have ... $= (v \sqcap -F * F) * (F \sqcap v)^\star$
by (*metis assms(1) equivalence-comp-right-complement mult-left-one star-one star-simulation-right-equal*)
also have ... $= (-F \sqcap v) * (F \sqcap v)^\star$
using *assms(1) equivalence-comp-right-complement inf.sup-monoid.add-commute* **by** *auto*
finally have $-F \sqcap v * F \leq (-F \sqcap v) * (F \sqcap v)^\star$
by *simp*
hence $(-F \sqcap v * F) * (-F \sqcap v * F)^T \leq (-F \sqcap v) * (F \sqcap v)^\star * ((-F \sqcap v) * (F \sqcap v)^\star)^T$
by (*simp add*: *comp-isotone conv-isotone*)
also have ... $= (-F \sqcap v) * (F \sqcap v)^\star * (F \sqcap v)^{T\star} * (-F \sqcap v)^T$
by (*simp add*: *comp-associative conv-dist-comp conv-star-commute*)
finally show *?thesis*
by *simp*
qed
also have ... $\leq (-F \sqcap v) * ((F \sqcap v^\star) \sqcup (F \sqcap v^{T\star})) * (-F \sqcap v)^T$
proof −

have $(F \sqcap v)^\star * (F \sqcap v)^{T\star} \leq F^\star * F^{T\star}$
using *fc-isotone* **by** *auto*
also have $... \leq F * F$
by (*metis assms*(*1*) *preorder-idempotent star.circ-sup-one-left-unfold star.circ-transitive-equal star-right-induct-mult*)
finally have *21*: $(F \sqcap v)^\star * (F \sqcap v)^{T\star} \leq F$
using *assms*(*1*) *dual-order.trans* **by** *blast*
have $(F \sqcap v)^\star * (F \sqcap v)^{T\star} \leq v^\star * v^{T\star}$
by (*simp add*: *fc-isotone*)
hence $(F \sqcap v)^\star * (F \sqcap v)^{T\star} \leq F \sqcap v^\star * v^{T\star}$
using *21* **by** *simp*
also have $... = F \sqcap (v^\star \sqcup v^{T\star})$
by (*simp add*: *assms*(*2*) *cancel-separate-eq*)
finally show *?thesis*
by (*metis assms*(*4*, *6*) *comp-associative comp-inf.semiring.distrib-left comp-isotone inf-pp-semi-commute mult-left-isotone regular-closed-inf*)
qed
also have $... \leq (-F \sqcap v) * (F \sqcap v^{T\star}) * (-F \sqcap v)^T \sqcup (-F \sqcap v) * (F \sqcap v^\star) * (-F \sqcap v)^T$
by (*simp add*: *mult-left-dist-sup mult-right-dist-sup*)
also have $... \leq (-F \sqcap v) * (-F \sqcap v)^T \sqcup (-F \sqcap v) * (-F \sqcap v)^T$
proof −
have $(-F \sqcap v) * (F \sqcap v^{T\star}) \leq (-F \sqcap v) * ((F \sqcap v)^{T\star} * (F \sqcap v)^\star \sqcap v^{T\star})$
by (*simp add*: *assms*(*5*) *inf.coboundedI1 mult-right-isotone*)
also have $... = (-F \sqcap v) * ((F \sqcap v)^T * (F \sqcap v)^{T\star} * (F \sqcap v)^\star \sqcap v^{T\star}) \sqcup (-F \sqcap v) * ((F \sqcap v)^\star \sqcap v^{T\star})$
by (*metis comp-associative comp-inf.mult-right-dist-sup mult-left-dist-sup star.circ-loop-fixpoint*)
also have $... \leq (-F \sqcap v) * (F \sqcap v)^T * top \sqcup (-F \sqcap v) * ((F \sqcap v)^\star \sqcap v^{T\star})$
by (*simp add*: *comp-associative comp-isotone inf.coboundedI2 inf.sup-monoid.add-commute le-supI1*)
also have $... \leq (-F \sqcap v) * (F \sqcap v)^T * top \sqcup (-F \sqcap v) * (v^\star \sqcap v^{T\star})$
by (*smt comp-inf.mult-right-isotone comp-inf.semiring.add-mono order.eq-iff inf.cobounded2 inf.sup-monoid.add-commute mult-right-isotone star-isotone*)
also have $... \leq bot \sqcup (-F \sqcap v) * (v^\star \sqcap v^{T\star})$
by (*metis assms*(*1*, *2*) *forests-bot-1 comp-associative comp-inf.semiring.add-right-mono mult-semi-associative vector-bot-closed*)
also have $... \leq -F \sqcap v$
by (*simp add*: *assms*(*2*) *acyclic-star-inf-conv*)
finally have *22*: $(-F \sqcap v) * (F \sqcap v^{T\star}) \leq -F \sqcap v$
by *simp*
have $((-F \sqcap v) * (F \sqcap v^{T\star}))^T = (F \sqcap v^\star) * (-F \sqcap v)^T$
by (*simp add*: *assms*(*1*) *conv-dist-inf conv-star-commute conv-dist-comp*)
hence $(F \sqcap v^\star) * (-F \sqcap v)^T \leq (-F \sqcap v)^T$
using *22 conv-isotone* **by** *fastforce*
thus *?thesis*
using *22* **by** (*metis assms*(*4*, *6*) *comp-associative comp-inf.pp-comp-semi-commute comp-inf.semiring.add-mono comp-isotone*

inf-pp-commute mult-left-isotone)
qed
also have ... = $(-F \sqcap v) * (-F \sqcap v)^T$
by *simp*
also have ... ≤ $v * v^T$
by (*simp add*: *comp-isotone conv-isotone*)
also have ... ≤ *1*
by (*simp add*: *assms*(*2*))
thus *?thesis*
using *calculation dual-order.trans* **by** *blast*
qed
have *3*: $top * ?i * top = top$
proof −
have *31*: *regular* $(e^T * -F * v * F * e)$
using *assms*(*3*, *4*, *6*) *arc-regular regular-mult-closed* **by** *auto*
have *32*: *bijective* $((top * e^T)^T)$
using *assms*(*3*) **by** (*simp add*: *conv-dist-comp*)
have $top * ?i * top = top * (v \sqcap -F * e * top) * ((top * e^T * F)^T \sqcap top)$
by (*simp add*: *comp-associative comp-inf-vector-1*)
also have ... = $(top \sqcap (-F * e * top)^T) * v * ((top * e^T * F)^T \sqcap top)$
using *comp-inf-vector conv-dist-comp* **by** *auto*
also have ... = $(-F * e * top)^T * v * (top * e^T * F)^T$
by *simp*
also have ... = $top^T * e^T * -F^T * v * F^T * e^{TT} * top^T$
by (*simp add*: *comp-associative conv-complement conv-dist-comp*)
finally have *33*: $top * ?i * top = top * e^T * -F * v * F * e * top$
by (*simp add*: *assms*(*1*))
have $top * ?i * top \neq bot$
proof (*rule ccontr*)
assume ¬ $top * (v \sqcap - F * e * top \sqcap top * e^T * F) * top \neq bot$
hence $top * e^T * -F * v * F * e * top = bot$
using *33* **by** *auto*
hence $e^T * -F * v * F * e = bot$
using *31 tarski comp-associative le-bot* **by** *fastforce*
hence $top * (-F * v * F * e)^T \leq -(e^T)$
by (*metis comp-associative conv-complement-sub-leq conv-involutive p-bot schroeder-5-p*)
hence $top * e^T * F^T * v^T * -F^T \leq -(e^T)$
by (*simp add*: *comp-associative conv-complement conv-dist-comp*)
hence $v * F * e * top * e^T \leq F$
by (*metis assms*(*1*, *4*) *comp-associative conv-dist-comp schroeder-3-p symmetric-top-closed*)
hence $v * F * e * top * top * e^T \leq F$
by (*simp add*: *comp-associative*)
hence $v * F * e * top \leq F * (top * e^T)^T$
using *32* **by** (*metis shunt-bijective comp-associative conv-involutive*)
hence $v * F * e * top \leq F * e * top$
using *comp-associative conv-dist-comp* **by** *auto*
hence $v^\star * F * e * top \leq F * e * top$

using *comp-associative star-left-induct-mult-iff* **by** *auto*
hence $e^T * F * e * top \leq F * e * top$
by (*meson assms*(*9*) *mult-left-isotone order-trans*)
hence $e^T * F * e * top * (e * top)^T \leq F$
using *32 shunt-bijective assms*(*3*) *mult-assoc* **by** *auto*
hence *34*: $e^T * F * e * top * top * e^T \leq F$
by (*metis conv-dist-comp mult.semigroup-axioms symmetric-top-closed semigroup.assoc*)
hence $e^T \leq F$
proof −
have $e^T \leq e^T * e * e^T$
by (*metis conv-involutive ex231c*)
also have ... $\leq e^T * F * e * e^T$
using *assms*(*1*) *comp-associative mult-left-isotone mult-right-isotone* **by** *fastforce*
also have ... $\leq e^T * F * e * top * top * e^T$
by (*simp add*: *mult-left-isotone top-right-mult-increasing vector-mult-closed*)
finally show *?thesis*
using *34* **by** *simp*
qed
hence *35*: $e \leq F$
using *assms*(*1*) *conv-order* **by** *fastforce*
have $top * (F * e)^T \leq - e$
using *assms*(*8*) *comp-associative schroeder-4-p* **by** *auto*
hence $top * e^T * F \leq - e$
by (*simp add*: *assms*(*1*) *comp-associative conv-dist-comp*)
hence $(top * e^T)^T * e \leq - F$
using *schroeder-3-p* **by** *auto*
hence $e * top * e \leq - F$
by (*simp add*: *conv-dist-comp*)
hence $e \leq - F$
by (*simp add*: *assms*(*3*) *arc-top-arc*)
hence $e \leq F \sqcap - F$
using *35 inf.boundedI* **by** *blast*
hence $e = bot$
using *bot-unique* **by** *auto*
thus *False*
using *assms*(*10*) **by** *auto*
qed
thus *?thesis*
by (*metis assms*(*3*, *4*, *6*) *arc-regular regular-closed-inf regular-closed-top regular-conv-closed regular-mult-closed semiring.mult-not-zero tarski*)
qed
have *bijective* (*?i* $* top$) $\wedge$ *bijective* (*?i*$^T * top$)
using *1 2 3 arc-expanded* **by** *blast*
thus *?thesis*
by *blast*
qed

4.4.3 Comparison of edge weights

In this section we compare the weight of the *selected-edge* with other edges of interest. For example, Theorem *e-leq-c-c-complement-transpose-general* is used to show that the *selected-edge* has its source inside and its target outside the component it is chosen for.

lemma *e-leq-c-c-complement-transpose-general*:
  assumes $e = minarc\ (v * -(v)^T \sqcap g)$
    and *regular v*
  shows $e \leq v * -(v)^T$
proof −
  have $e \leq -- (v * - v^T \sqcap g)$
    using *assms(1) minarc-below order-trans* **by** *blast*
  also have ... $\leq -- (v * - v^T)$
    using *order-lesseq-imp pp-isotone-inf* **by** *blast*
  also have ... $= v * - v^T$
    using *assms(2) regular-mult-closed* **by** *auto*
  finally show *?thesis*
    by *simp*
qed

lemma *x-leq-c-transpose-general*:
  assumes *vector-classes x v*
    and $a^T * top \leq x * e * top$
    and $e \leq v * -v^T$
  shows $a \leq v^T$
proof −
  have *1*: *equivalence x*
    using *assms(1)* **using** *vector-classes-def* **by** *blast*
  have $a \leq top * a$
    using *top-left-mult-increasing* **by** *blast*
  also have ... $\leq (x * e * top)^T$
    using *assms(2) conv-dist-comp conv-isotone* **by** *fastforce*
  also have ... $= top * e^T * x$
    using *1* **by** (*simp add*: *conv-dist-comp mult-assoc*)
  also have ... $\leq top * (v * - v^T)^T * x$
    by (*metis assms(3) conv-dist-comp conv-isotone mult-left-isotone symmetric-top-closed*)
  also have ... $= top * (- v * v^T) * x$
    by (*simp add*: *conv-complement conv-dist-comp*)
  also have ... $\leq top * v^T * x$
    by (*metis mult-left-isotone top.extremum mult-assoc*)
  also have ... $= v^T * x$
    using *assms(1) vector-classes-def vector-conv-covector* **by** *auto*
  also have ... $= v^T$
    by (*metis assms(1) order.antisym conv-dist-comp conv-order dual-order.trans mult-right-isotone mult-sub-right-one vector-classes-def*)
  finally show *?thesis*
    by *simp*

qed

lemma *x-leq-c-complement-general*:
assumes *vector v*
and $v * v^T \leq x$
and $a \leq v^T$
and $a \leq -x$
shows $a \leq -v$
proof −
have $a \leq -x \sqcap v^T$
using *assms*(*3*, *4*) **by** *auto*
also have ... $\leq -\ v$
proof −
have $v \sqcap v^T \leq x$
using *assms*(*1*, *2*) *vector-covector* **by** *auto*
hence $-x \sqcap v \sqcap v^T \leq bot$
using *inf.sup-monoid.add-assoc p-antitone pseudo-complement* **by** *fastforce*
thus *?thesis*
using *le-bot p-shunting-swap pseudo-complement* **by** *blast*
qed
finally show *?thesis*
by *simp*
qed

lemma *sum-e-below-sum-a-when-outgoing-same-component-general*:
assumes $e = minarc\ (v * -(v)^T \sqcap g)$
and *symmetric g*
and *arc a*
and $a \leq -x \sqcap --\ g$
and $a^T * top \leq x * e * top$
and *unique-vector-class x v*
shows $sum\ (e \sqcap g) \leq sum\ (a \sqcap g)$
proof −
have *1*:$e \leq v * -\ v^T$
using *assms*(*1*, *6*) *e-leq-c-c-complement-transpose-general*
unique-vector-class-def vector-classes-def **by** *auto*
have *2*: $a \leq v^T$
using *1 assms*(*5*) *assms*(*6*) *x-leq-c-transpose-general unique-vector-class-def*
by *blast*
hence $a \leq -v$
using *assms*(*4*, *6*) *inf.boundedE unique-vector-class-def vector-classes-def*
x-leq-c-complement-general **by** *meson*
hence $a \leq -\ v \sqcap v^T$
using *2* **by** *simp*
hence $a \leq -\ v * v^T$
using *assms*(*6*) *vector-complement-closed vector-covector*
unique-vector-class-def vector-classes-def **by** *metis*
hence $a^T \leq v^{TT} * -\ v^T$
using *conv-complement conv-dist-comp conv-isotone* **by** *metis*

hence *3*: $a^T \leq v * - v^T$
by *simp*
hence $a \leq -- g$
using *assms(4)* **by** *auto*
hence $a^T \leq -- g$
using *assms(2) conv-complement conv-isotone* **by** *fastforce*
hence $a^T \sqcap v * - v^T \sqcap -- g \neq bot$
using *3 assms(3, 6) inf.orderE semiring.mult-not-zero unique-vector-class-def vector-classes-def* **by** *metis*
hence $a^T \sqcap v * - v^T \sqcap g \neq bot$
using *inf.sup-monoid.add-commute pp-inf-bot-iff* **by** *auto*
hence *sum* $(minarc\ (v * - v^T \sqcap g) \sqcap (v * - v^T \sqcap g)) \leq sum\ (a^T \sqcap v * - v^T \sqcap g)$
using *assms(3) minarc-min inf.sup-monoid.add-assoc* **by** *simp*
hence *sum* $(e \sqcap v * - v^T \sqcap g) \leq sum\ (a^T \sqcap v * - v^T \sqcap g)$
using *assms(1, 6) inf.sup-monoid.add-assoc* **by** *simp*
hence *sum* $(e \sqcap g) \leq sum\ (a^T \sqcap g)$
using *1 3* **by** (*metis inf.orderE*)
hence *sum* $(e \sqcap g) \leq sum\ (a \sqcap g)$
by (*simp add*: *assms(2) sum-symmetric*)
thus *?thesis*
by *simp*
qed

lemma *sum-e-below-sum-x-when-outgoing-same-component*:
assumes *symmetric g*
and *vector j*
and *forest h*
and *regular h*
and $x \leq -$ *forest-components h* $\sqcap -- g$
and $x^T * top \leq$ *forest-components h* $*$ *selected-edge h j g* $* top$
and $j \neq bot$
and *arc x*
shows *sum* $(selected\text{-}edge\ h\ j\ g \sqcap g) \leq sum\ (x \sqcap g)$
proof −
let *?e* = *selected-edge h j g*
let *?c* = *choose-component* (*forest-components h*) *j*
let *?H* = *forest-components h*
show *?thesis*
proof (*rule sum-e-below-sum-a-when-outgoing-same-component-general*)
next
show $?e = minarc\ (?c * - ?c^T \sqcap g)$
by *simp*
next
show *symmetric g*
by (*simp add*: *assms(1)*)
next
show *arc x*
by (*simp add*: *assms(8)*)

```
  next
    show x ≤ -?H ⊓ -- g
      using assms(5) by auto
  next
    show x^T * top ≤ ?H * ?e * top
      by (simp add: assms(6))
  next
    show unique-vector-class ?H ?c
    proof (unfold unique-vector-class-def, unfold vector-classes-def, intro conjI)
    next
      show regular ?H
        by (metis assms(4) conv-complement pp-dist-star regular-mult-closed)
    next
      show regular ?c
        using component-is-regular by auto
    next
      show reflexive ?H
        using assms(3) forest-components-equivalence by blast
    next
      show transitive ?H
        using assms(3) fch-equivalence by blast
    next
      show symmetric ?H
        by (simp add: assms(3) fch-equivalence)
    next
      show vector ?c
        by (simp add: assms(2, 6) component-is-vector)
    next
      show ?H * ?c ≤ ?c
        using component-single by auto
    next
      show ?c ≠ bot
        using assms(2, 6 , 7, 8) inf-bot-left le-bot minarc-bot mult-left-zero
mult-right-zero by fastforce
    next
      show ?c * ?c^T ≤ ?H
        by (simp add: component-is-connected)
    qed
  qed
qed
```

If there is a path in the *forest-modulo-equivalence* from an edge between components, a, to the *selected-edge*, e, then the weight of e is no greater than the weight of a. This is because either,

* the edges a and e are adjacent the same component so that we can use *sum-e-below-sum-x-when-outgoing-same-component*, or

* there is at least one edge between a and e, namely x, the edge incoming to the component that e is outgoing from. The path from a to e is split

on x using *forest-modulo-equivalence-path-split-disj*. We show that the weight of e is no greater than the weight of x by making use of lemma *sum-e-below-sum-x-when-outgoing-same-component*. We define x in a way that we can show that the weight of x is no greater than the weight of a using the invariant. Then, it follows that the weight of e is no greater than the weight of a owing to transitivity.

lemma *a-to-e-in-forest-modulo-equivalence*:
assumes *symmetric g*
and $f \leq --g$
and *vector j*
and *forest h*
and *forest-modulo-equivalence* (*forest-components h*) *d*
and $f \sqcup f^T = h \sqcup h^T \sqcup d \sqcup d^T$
and $(\forall\ a\ b\ .$ *forest-modulo-equivalence-path a b* (*forest-components h*) $d \wedge a \leq -$(*forest-components h*) $\sqcap --\ g \wedge b \leq d \longrightarrow$ *sum*$(b \sqcap g) \leq$ *sum*$(a \sqcap g))$
and *regular d*
and $j \neq$ *bot*
and $b =$ *selected-edge h j g*
and *arc a*
and *forest-modulo-equivalence-path a b* (*forest-components h*) $(d \sqcup$ *selected-edge h j g*)
and $a \leq -$ *forest-components h* $\sqcap --\ g$
and *regular h*
shows *sum* $(b \sqcap g) \leq$ *sum* $(a \sqcap g)$
proof −
let *?p* = *path f h j g*
let *?e* = *selected-edge h j g*
let *?F* = *forest-components f*
let *?H* = *forest-components h*
have *sum* $(b \sqcap g) \leq$ *sum* $(a \sqcap g)$
proof (*cases* $a^T * top \leq$ *?H* $*$ *?e* $*$ *top*)
case *True*
show $a^T * top \leq$ *?H* $*$ *?e* $*$ *top* $\Longrightarrow$ *sum* $(b \sqcap g) \leq$ *sum* $(a \sqcap g)$
proof−
have *sum* $(?e \sqcap g) \leq$ *sum* $(a \sqcap g)$
proof (*rule sum-e-below-sum-x-when-outgoing-same-component*)
show *symmetric g*
using *assms*(*1*) **by** *auto*
next
show *vector j*
using *assms*(*3*) **by** *blast*
next
show *forest h*
by (*simp add*: *assms*(*4*))
next
show $a \leq -$ *?H* $\sqcap --\ g$
using *assms*(*13*) **by** *auto*
next

show $a^T * top \leq ?H * ?e * top$
using *True* **by** *auto*
next
show $j \neq bot$
by (*simp add*: *assms*(*9*))
next
show *arc a*
by (*simp add*: *assms*(*11*))
next
show *regular h*
using *assms*(*14*) **by** *auto*
qed
thus *?thesis*
using *assms*(*10*) **by** *auto*
qed
next
case *False*
show $\neg\ a^T * top \leq ?H * ?e * top \Longrightarrow sum\ (b \sqcap g) \leq sum\ (a \sqcap g)$
proof −
let $?d' = d \sqcup ?e$
let $?x = d \sqcap top * ?e^T * ?H \sqcap (?H * d^T)^\star * ?H * a^T * top$
have *61*: *arc* (*?x*)
proof (*rule shows-arc-x*)
show *forest-modulo-equivalence ?H d*
by (*simp add*: *assms*(*5*))
next
show *forest-modulo-equivalence-path a ?e ?H d*
proof −
have *611*: *forest-modulo-equivalence-path a b ?H* $(d \sqcup b)$
using *assms*(*10*, *12*) **by** *auto*
have *616*: *regular h*
using *assms*(*14*) **by** *auto*
have *regular a*
using *611 forest-modulo-equivalence-path-def arc-regular* **by** *fastforce*
thus *?thesis*
using *616* **by** (*smt forest-modulo-equivalence-path-split-disj assms*(*4*, *8*, *10*, *12*) *forest-modulo-equivalence-path-def fch-equivalence minarc-regular regular-closed-star regular-conv-closed regular-mult-closed*)
qed
next
show $(?H * d)^+ \leq -\ ?H$
using *assms*(*5*) *forest-modulo-equivalence-def le-bot pseudo-complement* **by** *blast*
next
show $\neg\ a^T * top \leq ?H * ?e * top$
by (*simp add*: *False*)
next
show *regular a*
using *assms*(*12*) *forest-modulo-equivalence-path-def arc-regular* **by** *auto*

next
show *regular ?e*
using *minarc-regular* **by** *auto*
next
show *regular ?H*
using *assms*(*14*) *pp-dist-star regular-conv-closed regular-mult-closed* **by** *auto*
next
show *regular d*
using *assms*(*8*) **by** *auto*
qed
have *62*: *bijective* (a^T ∗ *top*)
by (*simp add*: *assms*(*11*))
have *63*: *bijective* (*?x* ∗ *top*)
using *61* **by** *simp*
have *64*: *?x* ≤ (*?H* ∗ d^T)* ∗ *?H* ∗ a^T ∗ *top*
by *simp*
hence *?x* ∗ *top* ≤ (*?H* ∗ d^T)* ∗ *?H* ∗ a^T ∗ *top*
using *mult-left-isotone inf-vector-comp* **by** *auto*
hence a^T ∗ *top* ≤ ((*?H* ∗ d^T)* ∗ *?H*)^T ∗ *?x* ∗ *top*
using *62 63 64* **by** (*smt bijective-reverse mult-assoc*)
also have ... = *?H* ∗ (*d* ∗ *?H*)* ∗ *?x* ∗ *top*
using *conv-dist-comp conv-star-commute* **by** *auto*
also have ... = (*?H* ∗ *d*)* ∗ *?H* ∗ *?x* ∗ *top*
by (*simp add*: *star-slide*)
finally have a^T ∗ *top* ≤ (*?H* ∗ *d*)* ∗ *?H* ∗ *?x* ∗ *top*
by *simp*
hence *65*: *forest-modulo-equivalence-path a ?x ?H d*
using *61 assms*(*12*) *forest-modulo-equivalence-path-def* **by** *blast*
have *66*: *?x* ≤ *d*
by (*simp add*: *inf.sup-monoid.add-assoc*)
hence *x-below-a*: *sum* (*?x* ⊓ *g*) ≤ *sum* (*a* ⊓ *g*)
using *65 forest-modulo-equivalence-path-def assms*(*7*, *13*) **by** *blast*
have *sum* (*?e* ⊓ *g*) ≤ *sum* (*?x* ⊓ *g*)
proof (*rule sum-e-below-sum-x-when-outgoing-same-component*)
show *symmetric g*
using *assms*(*1*) **by** *auto*
next
show *vector j*
using *assms*(*3*) **by** *blast*
next
show *forest h*
by (*simp add*: *assms*(*4*))
next
show *?x* ≤ − *?H* ⊓ −− *g*
proof −
have *67*: *?x* ≤ − *?H*
proof −
have *?x* ≤ *d*

```
          using 66 by blast
        also have ... ≤ ?H ∗ d
          using dual-order.trans star.circ-loop-fixpoint sup.cobounded2
mult-assoc by metis
        also have ... ≤ (?H ∗ d)⁺
          using star.circ-mult-increasing by blast
        also have ... ≤ − ?H
          using assms(5) bot-unique pseudo-complement
forest-modulo-equivalence-def by blast
        thus ?thesis
          using calculation inf.order-trans by blast
      qed
      have ?x ≤ d
        by (simp add: conv-isotone inf.sup-monoid.add-assoc)
      also have ... ≤ f ⊔ f^T
      proof −
        have h ⊔ h^T ⊔ d ⊔ d^T = f ⊔ f^T
          by (simp add: assms(6))
        thus ?thesis
          by (metis (no-types) le-supE sup.absorb-iff2 sup.idem)
      qed
      also have ... ≤ −− g
        using assms(1, 2) conv-complement conv-isotone by fastforce
      finally have ?x ≤ −− g
        by simp
      thus ?thesis
        by (simp add: 67)
    qed
  next
    show ?x^T ∗ top ≤ ?H ∗ ?e ∗ top
    proof −
      have ?x ≤ top ∗ ?e^T ∗ ?H
        using inf.coboundedI1 by auto
      hence ?x^T ≤ ?H ∗ ?e ∗ top
        using conv-dist-comp conv-dist-inf conv-star-commute inf.orderI
inf.sup-monoid.add-assoc inf.sup-monoid.add-commute mult-assoc by auto
      hence ?x^T ∗ top ≤ ?H ∗ ?e ∗ top ∗ top
        by (simp add: mult-left-isotone)
      thus ?thesis
        by (simp add: mult-assoc)
    qed
  next
    show j ≠ bot
      by (simp add: assms(9))
  next
    show arc (?x)
      using 61 by blast
  next
    show regular h
```

```
      using assms(14) by auto
    qed
    hence sum (?e ⊓ g) ≤ sum (a ⊓ g)
      using x-below-a order.trans by blast
    thus ?thesis
      by (simp add: assms(10))
  qed
 qed
 thus ?thesis
  by simp
qed
```

4.4.4 Maintenance of algorithm invariants

In this section, most of the work is done to maintain the invariants of the inner and outer loops of the algorithm. In particular, we use *exists-a-w* to maintain that f can be extended to a minimum spanning forest.

```
lemma boruvka-exchange-spanning-inv:
 assumes forest v
  and v^⋆ ∗ e^T = e^T
  and i ≤ v ⊓ top ∗ e^T ∗ w^{T⋆}
  and arc i
  and arc e
  and v ≤ −−g
  and w ≤ −−g
  and e ≤ −−g
  and components g ≤ forest-components v
 shows i ≤ (v ⊓ −i)^{T⋆} ∗ e^T ∗ top
proof −
 have 1: (v ⊓ −i ⊓ −i^T) ∗ (v^T ⊓ −i ⊓ −i^T) ≤ 1
  using assms(1) comp-isotone order.trans inf.cobounded1 by blast
 have 2: bijective (i ∗ top) ∧ bijective (e^T ∗ top)
  using assms(4, 5) mult-assoc by auto
 have i ≤ v ∗ (top ∗ e^T ∗ w^{T⋆})^T
  using assms(3) covector-mult-closed covector-restrict-comp-conv
order-lesseq-imp vector-top-closed by blast
 also have ... ≤ v ∗ w^{T⋆T} ∗ e^{TT} ∗ top^T
  by (simp add: comp-associative conv-dist-comp)
 also have ... ≤ v ∗ w^⋆ ∗ e ∗ top
  by (simp add: conv-star-commute)
 also have ... = v ∗ w^⋆ ∗ e ∗ e^T ∗ e ∗ top
  using assms(5) arc-eq-1 by (simp add: comp-associative)
 also have ... ≤ v ∗ w^⋆ ∗ e ∗ e^T ∗ top
  by (simp add: comp-associative mult-right-isotone)
 also have ... ≤ (−−g) ∗ (−−g)^⋆ ∗ (−−g) ∗ e^T ∗ top
  using assms(6, 7, 8) by (simp add: comp-isotone star-isotone)
 also have ... ≤ (−−g)^⋆ ∗ e^T ∗ top
  by (metis comp-isotone mult-left-isotone star.circ-increasing
star.circ-transitive-equal)
```

also have ... $\leq v^{T\star} * v^\star * e^T * top$
by (*simp add*: *assms*(*9*) *mult-left-isotone*)
also have ... $\leq v^{T\star} * e^T * top$
by (*simp add*: *assms*(*2*) *comp-associative*)
finally have $i \leq v^{T\star} * e^T * top$
by *simp*
hence $i * top \leq v^{T\star} * e^T * top$
by (*metis comp-associative mult-left-isotone vector-top-closed*)
hence $e^T * top \leq v^{T\star T} * i * top$
using *2* **by** (*metis bijective-reverse mult-assoc*)
also have ... $= v^\star * i * top$
by (*simp add*: *conv-star-commute*)
also have ... $\leq (v \sqcap -i \sqcap -i^T)^\star * i * top$
proof −
have *3*: $i * top \leq (v \sqcap -i \sqcap -i^T)^\star * i * top$
using *star.circ-loop-fixpoint sup-right-divisibility mult-assoc* **by** *auto*
have $(v \sqcap i) * (v \sqcap -i \sqcap -i^T)^\star * i * top \leq i * top * i * top$
by (*metis comp-isotone inf.cobounded1 inf.sup-monoid.add-commute mult-left-isotone top.extremum*)
also have ... $\leq i * top$
by *simp*
finally have *4*: $(v \sqcap i) * (v \sqcap -i \sqcap -i^T)^\star * i * top \leq (v \sqcap -i \sqcap -i^T)^\star * i * top$
using *3 dual-order.trans* **by** *blast*
have *5*: $(v \sqcap -i \sqcap -i^T) * (v \sqcap -i \sqcap -i^T)^\star * i * top \leq (v \sqcap -i \sqcap -i^T)^\star * i * top$
by (*metis mult-left-isotone star.circ-increasing star.left-plus-circ*)
have $v^+ \leq -1$
by (*simp add*: *assms*(*1*))
hence $v * v \leq -1$
by (*metis mult-left-isotone order-trans star.circ-increasing star.circ-plus-same*)
hence $v * 1 \leq -v^T$
by (*simp add*: *schroeder-5-p*)
hence $v \leq -v^T$
by *simp*
hence $v \sqcap v^T \leq bot$
by (*simp add*: *bot-unique pseudo-complement*)
hence *7*: $v \sqcap i^T \leq bot$
by (*metis assms*(*3*) *comp-inf.mult-right-isotone conv-dist-inf inf.boundedE inf.le-iff-sup le-bot*)
hence $(v \sqcap i^T) * (v \sqcap -i \sqcap -i^T)^\star * i * top \leq bot$
using *le-bot semiring.mult-zero-left* **by** *fastforce*
hence *6*: $(v \sqcap i^T) * (v \sqcap -i \sqcap -i^T)^\star * i * top \leq (v \sqcap -i \sqcap -i^T)^\star * i * top$
using *bot-least le-bot* **by** *blast*
have *8*: $v = (v \sqcap i) \sqcup (v \sqcap i^T) \sqcup (v \sqcap -i \sqcap -i^T)$
proof −
have *81*: *regular i*
by (*simp add*: *assms*(*4*) *arc-regular*)

have $(v \sqcap i^T) \sqcup (v \sqcap -i \sqcap -i^T) = (v \sqcap -i)$
using *7* **by** (*metis comp-inf.coreflexive-comp-inf-complement inf-import-p inf-p le-bot maddux-3-11-pp top.extremum*)
hence $(v \sqcap i) \sqcup (v \sqcap i^T) \sqcup (v \sqcap -i \sqcap -i^T) = (v \sqcap i) \sqcup (v \sqcap -i)$
by (*simp add*: *sup.semigroup-axioms semigroup.assoc*)
also have ... $= v$
using *81* **by** (*metis maddux-3-11-pp*)
finally show *?thesis*
by *simp*
qed
have $(v \sqcap i) * (v \sqcap -i \sqcap -i^T)^\star * i * top \sqcup (v \sqcap i^T) * (v \sqcap -i \sqcap -i^T)^\star * i * top \sqcup (v \sqcap -i \sqcap -i^T) * (v \sqcap -i \sqcap -i^T)^\star * i * top \leq (v \sqcap -i \sqcap -i^T)^\star * i * top$
using *4 5 6* **by** *simp*
hence $((v \sqcap i) \sqcup (v \sqcap i^T) \sqcup (v \sqcap -i \sqcap -i^T)) * (v \sqcap -i \sqcap -i^T)^\star * i * top \leq (v \sqcap -i \sqcap -i^T)^\star * i * top$
by (*simp add*: *mult-right-dist-sup*)
hence $v * (v \sqcap -i \sqcap -i^T)^\star * i * top \leq (v \sqcap -i \sqcap -i^T)^\star * i * top$
using *8* **by** *auto*
hence $i * top \sqcup v * (v \sqcap -i \sqcap -i^T)^\star * i * top \leq (v \sqcap -i \sqcap -i^T)^\star * i * top$
using *3* **by** *auto*
hence *9*:$v^\star * (v \sqcap -i \sqcap -i^T)^\star * i * top \leq (v \sqcap -i \sqcap -i^T)^\star * i * top$
by (*simp add*: *star-left-induct-mult mult-assoc*)
have $v^\star * i * top \leq v^\star * (v \sqcap -i \sqcap -i^T)^\star * i * top$
using *3 mult-right-isotone mult-assoc* **by** *auto*
thus *?thesis*
using *9 order.trans* **by** *blast*
qed
finally have $e^T * top \leq (v \sqcap -i \sqcap -i^T)^\star * i * top$
by *simp*
hence $i * top \leq (v \sqcap -i \sqcap -i^T)^{\star T} * e^T * top$
using *2* **by** (*metis bijective-reverse mult-assoc*)
also have ... $= (v^T \sqcap -i \sqcap -i^T)^\star * e^T * top$
using *comp-inf.inf-vector-comp conv-complement conv-dist-inf conv-star-commute inf.sup-monoid.add-commute* **by** *auto*
also have ... $\leq ((v \sqcap -i \sqcap -i^T) \sqcup (v^T \sqcap -i \sqcap -i^T))^\star * e^T * top$
by (*simp add*: *mult-left-isotone star-isotone*)
finally have $i \leq ((v^T \sqcap -i \sqcap -i^T) \sqcup (v \sqcap -i \sqcap -i^T))^\star * e^T * top$
using *dual-order.trans top-right-mult-increasing sup-commute* **by** *auto*
also have ... $= (v^T \sqcap -i \sqcap -i^T)^\star * (v \sqcap -i \sqcap -i^T)^\star * e^T * top$
using *1 cancel-separate-1* **by** (*simp add*: *sup-commute*)
also have ... $\leq (v^T \sqcap -i \sqcap -i^T)^\star * v^\star * e^T * top$
by (*simp add*: *inf-assoc mult-left-isotone mult-right-isotone star-isotone*)
also have ... $= (v^T \sqcap -i \sqcap -i^T)^\star * e^T * top$
using *assms(2) mult-assoc* **by** *simp*
also have ... $\leq (v^T \sqcap -i^T)^\star * e^T * top$
by (*metis mult-left-isotone star-isotone inf.cobounded2 inf.left-commute inf.sup-monoid.add-commute*)
also have ... $= (v \sqcap -i)^{T\star} * e^T * top$
using *conv-complement conv-dist-inf* **by** *auto*

finally show *?thesis*
by *simp*
qed

lemma *exists-a-w*:
assumes *symmetric g*
and *forest f*
and $f \leq --g$
and *regular f*
and $(\exists w\ .\ \textit{minimum-spanning-forest}\ w\ g \wedge f \leq w \sqcup w^T)$
and *vector j*
and *regular j*
and *forest h*
and *forest-modulo-equivalence* (*forest-components h*) *d*
and $d * \textit{top} \leq - j$
and *forest-components h* $* j = j$
and $f \sqcup f^T = h \sqcup h^T \sqcup d \sqcup d^T$
and $(\forall\ a\ b\ .\ \textit{forest-modulo-equivalence-path}\ a\ b\ (\textit{forest-components}\ h)\ d \wedge a \leq -(\textit{forest-components}\ h) \sqcap -- g \wedge b \leq d \longrightarrow \textit{sum}(b \sqcap g) \leq \textit{sum}(a \sqcap g))$
and *regular d*
and *selected-edge h j g* $\leq -$ *forest-components f*
and *selected-edge h j g* $\neq$ *bot*
and $j \neq \textit{bot}$
and *regular h*
and $h \leq --g$
shows $\exists w.$ *minimum-spanning-forest w g* $\wedge$
$f \sqcap - (\textit{selected-edge}\ h\ j\ g)^T \sqcap - (\textit{path}\ f\ h\ j\ g) \sqcup (f \sqcap - (\textit{selected-edge}\ h\ j\ g)^T \sqcap (\textit{path}\ f\ h\ j\ g))^T \sqcup (\textit{selected-edge}\ h\ j\ g) \leq w \sqcup w^T$
proof −
let *?p* = *path f h j g*
let *?e* = *selected-edge h j g*
let $?f = (f \sqcap -?e^T \sqcap -?p) \sqcup (f \sqcap -?e^T \sqcap ?p)^T \sqcup ?e$
let *?F* = *forest-components f*
let *?H* = *forest-components h*
let *?ec* = *choose-component* (*forest-components h*) *j* $*$ − *choose-component* (*forest-components h*) $j^T \sqcap g$
from *assms*(*4*) **obtain** *w* **where** *2*: *minimum-spanning-forest w g* $\wedge$ $f \leq w \sqcup w^T$
using *assms*(*5*) **by** *blast*
hence *3*: *regular w* $\wedge$ *regular f* $\wedge$ *regular ?e*
by (*metis assms*(*4*) *minarc-regular minimum-spanning-forest-def spanning-forest-def*)
have *5*: *equivalence ?F*
using *assms*(*2*) *forest-components-equivalence* **by** *auto*
have $?e^T * \textit{top} * ?e^T = ?e^T$
by (*metis arc-conv-closed arc-top-arc coreflexive-bot-closed coreflexive-symmetric minarc-arc minarc-bot-iff semiring.mult-not-zero*)
hence $?e^T * \textit{top} * ?e^T \leq -?F$
using *5 assms*(*15*) *conv-complement conv-isotone* **by** *fastforce*

hence *6*: $?e * ?F * ?e = bot$
using *assms(2) le-bot triple-schroeder-p* **by** *simp*
let $?q = w \sqcap top * ?e * w^{T\star}$
let $?v = (w \sqcap -(top * ?e * w^{T\star})) \sqcup ?q^T$
have *7*: *regular ?q*
using *3 regular-closed-star regular-conv-closed regular-mult-closed* **by** *auto*
have *8*: *injective ?v*
proof (*rule kruskal-exchange-injective-inv-1*)
show *injective w*
using *2 minimum-spanning-forest-def spanning-forest-def* **by** *blast*
next
show *covector* $(top * ?e * w^{T\star})$
by (*simp add*: *covector-mult-closed*)
next
show $top * ?e * w^{T\star} * w^T \leq top * ?e * w^{T\star}$
by (*simp add*: *mult-right-isotone star.right-plus-below-circ mult-assoc*)
next
show *coreflexive* $((top * ?e * w^{T\star})^T * (top * ?e * w^{T\star}) \sqcap w^T * w)$
using *2* **by** (*metis comp-inf.semiring.mult-not-zero forest-bot kruskal-injective-inv-3 minarc-arc minarc-bot-iff minimum-spanning-forest-def semiring.mult-not-zero spanning-forest-def*)
qed
have *9*: *components g* $\leq$ *forest-components ?v*
proof (*rule kruskal-exchange-spanning-inv-1*)
show *injective* $(w \sqcap - (top *?e * w^{T\star}) \sqcup (w \sqcap top * ?e * w^{T\star})^T)$
using *8* **by** *simp*
next
show *regular* $(w \sqcap top * ?e * w^{T\star})$
using *7* **by** *simp*
next
show *components g* $\leq$ *forest-components w*
using *2 minimum-spanning-forest-def spanning-forest-def* **by** *blast*
qed
have *10*: *spanning-forest ?v g*
proof (*unfold spanning-forest-def*, *intro conjI*)
show *injective ?v*
using *8* **by** *auto*
next
show *acyclic ?v*
proof (*rule kruskal-exchange-acyclic-inv-1*)
show *pd-kleene-allegory-class.acyclic w*
using *2 minimum-spanning-forest-def spanning-forest-def* **by** *blast*
next
show *covector* $(top * ?e * w^{T\star})$
by (*simp add*: *covector-mult-closed*)
qed
next
show $?v \leq --g$
proof (*rule sup-least*)

```
    show w ⊓ − (top ∗ ?e ∗ w^T⋆) ≤ − − g
      using 7 inf.coboundedI1 minimum-spanning-forest-def spanning-forest-def 2
by blast
    next
      show (w ⊓ top ∗ ?e ∗ w^T⋆)^T ≤ − − g
        using 2 by (metis assms(1) conv-complement conv-isotone
inf.coboundedI1 minimum-spanning-forest-def spanning-forest-def)
    qed
  next
    show components g ≤ forest-components ?v
      using 9 by simp
  next
    show regular ?v
      using 3 regular-closed-star regular-conv-closed regular-mult-closed by auto
  qed
  have 11: sum (?v ⊓ g) = sum (w ⊓ g)
  proof −
    have sum (?v ⊓ g) = sum (w ⊓ −(top ∗ ?e ∗ w^T⋆) ⊓ g) + sum (?q^T ⊓ g)
      using 2 by (smt conv-complement conv-top epm-8 inf-import-p inf-top-right
regular-closed-top vector-top-closed minimum-spanning-forest-def
spanning-forest-def sum-disjoint)
    also have ... = sum (w ⊓ −(top ∗ ?e ∗ w^T⋆) ⊓ g) + sum (?q ⊓ g)
      by (simp add: assms(1) sum-symmetric)
    also have ... = sum (((w ⊓ −(top ∗ ?e ∗ w^T⋆)) ⊔ ?q) ⊓ g)
      using inf-commute inf-left-commute sum-disjoint by simp
    also have ... = sum (w ⊓ g)
      using 3 7 8 maddux-3-11-pp by auto
    finally show ?thesis
      by simp
  qed
  have 12: ?v ⊔ ?v^T = w ⊔ w^T
  proof −
    have ?v ⊔ ?v^T = (w ⊓ −?q) ⊔ ?q^T ⊔ (w^T ⊓ −?q^T) ⊔ ?q
      using conv-complement conv-dist-inf conv-dist-sup inf-import-p sup-assoc by
simp
    also have ... = w ⊔ w^T
      using 3 7 conv-complement conv-dist-inf inf-import-p maddux-3-11-pp
sup-monoid.add-assoc sup-monoid.add-commute by auto
    finally show ?thesis
      by simp
  qed
  have 13: ?v ∗ ?e^T = bot
  proof (rule kruskal-reroot-edge)
    show injective (?e^T ∗ top)
      using assms(16) minarc-arc minarc-bot-iff by blast
  next
    show pd-kleene-allegory-class.acyclic w
      using 2 minimum-spanning-forest-def spanning-forest-def by simp
  qed
```

have *?v* ⊓ *?e* ≤ *?v* ⊓ *top* ∗ *?e*
using *inf.sup-right-isotone top-left-mult-increasing* **by** *simp*
also have ... ≤ *?v* ∗ (*top* ∗ *?e*)^T
using *covector-restrict-comp-conv covector-mult-closed vector-top-closed* **by** *simp*
finally have *14*: *?v* ⊓ *?e* = *bot*
using *13* **by** (*metis conv-dist-comp mult-assoc le-bot mult-left-zero*)
let *?i* = *?v* ⊓ (− *?F*) ∗ *?e* ∗ *top* ⊓ *top* ∗ *?e*^{T} ∗ *?F*
let *?w* = (*?v* ⊓ −*?i*) ⊔ *?e*
have *15*: *regular ?i*
using *3 regular-closed-star regular-conv-closed regular-mult-closed* **by** *simp*
have *16*: *?F* ≤ −*?i*
proof −
have *161*: *bijective* (*?e* ∗ *top*)
using *assms*(*16*) *minarc-arc minarc-bot-iff* **by** *auto*
have *?i* ≤ − *?F* ∗ *?e* ∗ *top*
using *inf.cobounded2 inf.coboundedI1* **by** *blast*
also have ... = − (*?F* ∗ *?e* ∗ *top*)
using *161 comp-bijective-complement* **by** (*simp add*: *mult-assoc*)
finally have *?i* ≤ − (*?F* ∗ *?e* ∗ *top*)
by *blast*
hence *162*: *?i* ⊓ *?F* ≤ − (*?F* ∗ *?e* ∗ *top*)
using *inf.coboundedI1* **by** *blast*
have *?i* ⊓ *?F* ≤ *?F* ⊓ (*top* ∗ *?e*^{T} ∗ *?F*)
by (*meson inf-le1 inf-le2 le-infI order-trans*)
also have ... ≤ *?F* ∗ (*top* ∗ *?e*^{T} ∗ *?F*)^T
by (*simp add*: *covector-mult-closed covector-restrict-comp-conv*)
also have ... = *?F* ∗ *?F*^{T} ∗ *?e*^{TT} ∗ *top*^{T}
by (*simp add*: *conv-dist-comp mult-assoc*)
also have ... = *?F* ∗ *?F* ∗ *?e* ∗ *top*
by (*simp add*: *conv-dist-comp conv-star-commute*)
also have ... = *?F* ∗ *?e* ∗ *top*
by (*simp add*: *5 preorder-idempotent*)
finally have *?i* ⊓ *?F* ≤ *?F* ∗ *?e* ∗ *top*
by *simp*
hence *?i* ⊓ *?F* ≤ *?F* ∗ *?e* ∗ *top* ⊓ − (*?F* ∗ *?e* ∗ *top*)
using *162 inf.bounded-iff* **by** *blast*
also have ... = *bot*
by *simp*
finally show *?thesis*
using *le-bot p-antitone-iff pseudo-complement* **by** *blast*
qed
have *17*: *?i* ≤ *top* ∗ *?e*^{T} ∗ (*?F* ⊓ *?v* ⊓ −*?i*)^{T*}
proof −
have *?i* ≤ *?v* ⊓ − *?F* ∗ *?e* ∗ *top* ⊓ *top* ∗ *?e*^{T} ∗ (*?F* ⊓ *?v*)^{T*} ∗ (*?F* ⊓ *?v*)*
using *2 8 12* **by** (*smt inf.sup-right-isotone kruskal-forest-components-inf mult-right-isotone mult-assoc*)
also have ... = *?v* ⊓ − *?F* ∗ *?e* ∗ *top* ⊓ *top* ∗ *?e*^{T} ∗ (*?F* ⊓ *?v*)^{T*} ∗ (*1* ⊔ (*?F* ⊓ *?v*)* ∗ (*?F* ⊓ *?v*))

using *star-left-unfold-equal star.circ-right-unfold-1* **by** *auto*
also have ... = $?v \sqcap - ?F * ?e * top \sqcap (top * ?e^T * (?F \sqcap ?v)^{T\star} \sqcup top * ?e^T * (?F \sqcap ?v)^{T\star} * (?F \sqcap ?v)^\star * (?F \sqcap ?v))$
by (*simp add*: *mult-left-dist-sup mult-assoc*)
also have ... = $(?v \sqcap - ?F * ?e * top \sqcap top * ?e^T * (?F \sqcap ?v)^{T\star}) \sqcup (?v \sqcap - ?F * ?e * top \sqcap top * ?e^T * (?F \sqcap ?v)^{T\star} * (?F \sqcap ?v)^\star * (?F \sqcap ?v))$
using *comp-inf.semiring.distrib-left* **by** *blast*
also have ... $\leq top * ?e^T * (?F \sqcap ?v)^{T\star} \sqcup (?v \sqcap - ?F * ?e * top \sqcap top * ?e^T * (?F \sqcap ?v)^{T\star} * (?F \sqcap ?v)^\star * (?F \sqcap ?v))$
using *comp-inf.semiring.add-right-mono inf-le2* **by** *blast*
also have ... $\leq top * ?e^T * (?F \sqcap ?v)^{T\star} \sqcup (?v \sqcap - ?F * ?e * top \sqcap top * ?e^T * (?F^T \sqcap ?v^T)^\star * (?F \sqcap ?v)^\star * (?F \sqcap ?v))$
by (*simp add*: *conv-dist-inf*)
also have ... $\leq top * ?e^T * (?F \sqcap ?v)^{T\star} \sqcup (?v \sqcap - ?F * ?e * top \sqcap top * ?e^T * ?F^{T\star} * ?F^\star * (?F \sqcap ?v))$
proof −
have $top * ?e^T * (?F^T \sqcap ?v^T)^\star * (?F \sqcap ?v)^\star * (?F \sqcap ?v) \leq top * ?e^T * ?F^{T\star} * ?F^\star * (?F \sqcap ?v)$
using *star-isotone* **by** (*simp add*: *comp-isotone*)
hence $?v \sqcap - ?F * ?e * top \sqcap top * ?e^T * (?F^T \sqcap ?v^T)^\star * (?F \sqcap ?v)^\star * (?F \sqcap ?v) \leq ?v \sqcap - ?F * ?e * top \sqcap top * ?e^T * ?F^{T\star} * ?F^\star * (?F \sqcap ?v)$
using *inf.sup-right-isotone* **by** *blast*
thus *?thesis*
using *sup-right-isotone* **by** *blast*
qed
also have ... = $top * ?e^T * (?F \sqcap ?v)^{T\star} \sqcup (?v \sqcap - ?F * ?e * top \sqcap top * ?e^T * ?F^\star * ?F^\star * (?F \sqcap ?v))$
using *5* **by** *auto*
also have ... = $top * ?e^T * (?F \sqcap ?v)^{T\star} \sqcup (?v \sqcap - ?F * ?e * top \sqcap top * ?e^T * ?F * ?F * (?F \sqcap ?v))$
by (*simp add*: *assms(2) forest-components-star*)
also have ... = $top * ?e^T * (?F \sqcap ?v)^{T\star} \sqcup (?v \sqcap - ?F * ?e * top \sqcap top * ?e^T * ?F * (?F \sqcap ?v))$
using *5 mult.semigroup-axioms preorder-idempotent semigroup.assoc* **by** *fastforce*
also have ... = $top * ?e^T * (?F \sqcap ?v)^{T\star}$
proof −
have $?e * top * ?e^T \leq 1$
using *assms(16) arc-expanded minarc-arc minarc-bot-iff* **by** *auto*
hence $?F * ?e * top * ?e^T \leq ?F * 1$
by (*metis comp-associative comp-isotone mult-semi-associative star.circ-transitive-equal*)
hence $?v * ?v^T * ?F * ?e * top * ?e^T \leq 1 * ?F * 1$
using *8* **by** (*smt comp-isotone mult-assoc*)
hence *171*: $?v * ?v^T * ?F * ?e * top * ?e^T \leq ?F$
by *simp*
hence $?v * (?F \sqcap ?v)^T * ?F * ?e * top * ?e^T \leq ?F$
proof −
have $?v * (?F \sqcap ?v)^T * ?F * ?e * top * ?e^T \leq ?v * ?v^T * ?F * ?e * top *$

$?e^T$
by (*simp add*: *conv-dist-inf mult-left-isotone mult-right-isotone*)
thus *?thesis*
using *171 order-trans* **by** *blast*
qed
hence *172*: $-?F * ((?F \sqcap ?v)^T * ?F * ?e * top * ?e^T)^T \leq -?v$
by (*smt schroeder-4-p comp-associative order-lesseq-imp pp-increasing*)
have $-?F * ((?F \sqcap ?v)^T * ?F * ?e * top * ?e^T)^T = -?F * ?e^{TT} * top^T * ?e^T * ?F^T * (?F \sqcap ?v)^{TT}$
by (*simp add*: *comp-associative conv-dist-comp*)
also have $... = -?F * ?e * top * ?e^T * ?F * (?F \sqcap ?v)$
using *5* **by** *auto*
also have $... = -?F * ?e * top * top * ?e^T * ?F * (?F \sqcap ?v)$
using *comp-associative* **by** *auto*
also have $... = -?F * ?e * top \sqcap top * ?e^T * ?F * (?F \sqcap ?v)$
by (*smt comp-associative comp-inf.star.circ-decompose-9 comp-inf.star-star-absorb comp-inf-covector inf-vector-comp vector-top-closed*)
finally have $-?F * ((?F \sqcap ?v)^T * ?F * ?e * top * ?e^T)^T = -?F * ?e * top \sqcap top * ?e^T * ?F * (?F \sqcap ?v)$
by *simp*
hence $-?F * ?e * top \sqcap top * ?e^T * ?F * (?F \sqcap ?v) \leq -?v$
using *172* **by** *auto*
hence $?v \sqcap -?F * ?e * top \sqcap top * ?e^T * ?F * (?F \sqcap ?v) \leq bot$
by (*smt bot-unique inf.sup-monoid.add-commute p-shunting-swap pseudo-complement*)
thus *?thesis*
using *le-bot sup-monoid.add-0-right* **by** *blast*
qed
also have $... = top * ?e^T * (?F \sqcap ?v \sqcap -?i)^{T\star}$
using *16* **by** (*smt comp-inf.coreflexive-comp-inf-complement inf-top-right p-bot pseudo-complement top.extremum*)
finally show *?thesis*
by *blast*
qed
have *18*: $?i \leq top * ?e^T * ?w^{T\star}$
proof −
have $?i \leq top * ?e^T * (?F \sqcap ?v \sqcap -?i)^{T\star}$
using *17* **by** *simp*
also have $... \leq top * ?e^T * (?v \sqcap -?i)^{T\star}$
using *mult-right-isotone conv-isotone star-isotone inf.cobounded2 inf.sup-monoid.add-assoc* **by** (*simp add*: *inf.sup-monoid.add-assoc order.eq-iff inf.sup-monoid.add-commute*)
also have $... \leq top * ?e^T * ((?v \sqcap -?i) \sqcup ?e)^{T\star}$
using *mult-right-isotone conv-isotone star-isotone sup-ge1* **by** *simp*
finally show *?thesis*
by *blast*
qed
have *19*: $?i \leq top * ?e^T * ?v^{T\star}$
proof −

have *?i* $\leq$ *top* $*$ *?e*^{T} $*$ (*?F* $\sqcap$ *?v* $\sqcap$ $-$*?i*)^{T*}
using *17* **by** *simp*
also have ... $\leq$ *top* $*$ *?e*^{T} $*$ (*?v* $\sqcap$ $-$*?i*)^{T*}
using *mult-right-isotone conv-isotone star-isotone inf.cobounded2 inf.sup-monoid.add-assoc* **by** (*simp add*: *inf.sup-monoid.add-assoc order.eq-iff inf.sup-monoid.add-commute*)
also have ... $\leq$ *top* $*$ *?e*^{T} $*$ (*?v*)^{T*}
using *mult-right-isotone conv-isotone star-isotone* **by** *auto*
finally show *?thesis*
by *blast*
qed
have *20*: *f* $\sqcup$ *f*^{T} $\leq$ (*?v* $\sqcap$ $-$*?i* $\sqcap$ $-$*?i*^{T}) $\sqcup$ (*?v*^{T} $\sqcap$ $-$*?i* $\sqcap$ $-$*?i*^{T})
proof (*rule kruskal-edge-between-components-2*)
show *?F* $\leq$ $-$ *?i*
using *16* **by** *simp*
next
show *injective f*
by (*simp add*: *assms*(*2*))
next
show *f* $\sqcup$ *f*^{T} $\leq$ *w* $\sqcap$ $-$ (*top* $*$ *?e* $*$ *w*^{T*}) $\sqcup$ (*w* $\sqcap$ *top* $*$ *?e* $*$ *w*^{T*})^T $\sqcup$ (*w* $\sqcap$ $-$ (*top* $*$ *?e* $*$ *w*^{T*}) $\sqcup$ (*w* $\sqcap$ *top* $*$ *?e* $*$ *w*^{T*})^T)^T
using *2 12* **by** (*metis conv-dist-sup conv-involutive conv-isotone le-supI sup-commute*)
qed
have *minimum-spanning-forest ?w g* $\wedge$ *?f* $\leq$ *?w* $\sqcup$ *?w*^{T}
proof (*intro conjI*)
have *211*: *?e*^{T} $\leq$ *?v**
proof (*rule kruskal-edge-arc-1*[**where** *g*=*g* **and** *h*=*?ec*])
show *?e* $\leq$ $--$ *?ec*
using *minarc-below* **by** *blast*
next
show *?ec* $\leq$ *g*
using *assms*(*4*) *inf.cobounded2* **by** (*simp add*: *boruvka-inner-invariant-def boruvka-outer-invariant-def conv-dist-inf*)
next
show *symmetric g*
by (*meson assms*(*1*) *boruvka-inner-invariant-def boruvka-outer-invariant-def*)
next
show *components g* $\leq$ *forest-components* (*w* $\sqcap$ $-$ (*top* $*$ *?e* $*$ *w*^{T*}) $\sqcup$ (*w* $\sqcap$ *top* $*$ *?e* $*$ *w*^{T*})^T)
using *9* **by** *simp*
next
show (*w* $\sqcap$ $-$ (*top* $*$ *?e* $*$ *w*^{T*}) $\sqcup$ (*w* $\sqcap$ *top* $*$ *?e* $*$ *w*^{T*})^T) $*$ *?e*^{T} = *bot*
using *13* **by** *blast*
qed
have *212*: *arc ?i*
proof (*rule boruvka-edge-arc*)
show *equivalence ?F*

by (*simp add*: *5*)
next
show *forest ?v*
using *10 spanning-forest-def* **by** *blast*
next
show *arc ?e*
using *assms*(*16*) *minarc-arc minarc-bot-iff* **by** *blast*
next
show *regular ?F*
using *3 regular-closed-star regular-conv-closed regular-mult-closed* **by** *auto*
next
show $?F \leq$ *forest-components* ($?F \sqcap ?v$)
by (*simp add*: *12 2 8 kruskal-forest-components-inf*)
next
show *regular ?v*
using *10 spanning-forest-def* **by** *blast*
next
show $?v * ?e^T = bot$
using *13* **by** *auto*
next
show $?e * ?F * ?e = bot$
by (*simp add*: *6*)
next
show $?e^T \leq ?v^\star$
using *211* **by** *auto*

next
show $?e \neq bot$
by (*simp add*: *assms*(*16*))
qed
show *minimum-spanning-forest ?w g*
proof (*unfold minimum-spanning-forest-def*, *intro conjI*)
have $(?v \sqcap -?i) * ?e^T \leq ?v * ?e^T$
using *inf-le1 mult-left-isotone* **by** *simp*
hence $(?v \sqcap -?i) * ?e^T = bot$
using *13 le-bot* **by** *simp*
hence *221*: $?e * (?v \sqcap -?i)^T = bot$
using *conv-dist-comp conv-involutive conv-bot* **by** *force*
have *222*: *injective ?w*
proof (*rule injective-sup*)
show *injective* ($?v \sqcap -?i$)
using *8* **by** (*simp add*: *injective-inf-closed*)
next
show *coreflexive* ($?e * (?v \sqcap -?i)^T$)
using *221* **by** *simp*
next
show *injective ?e*
by (*metis arc-injective minarc-arc coreflexive-bot-closed*
coreflexive-injective minarc-bot-iff)

qed
show *spanning-forest ?w g*
proof (*unfold spanning-forest-def*, *intro conjI*)
show *injective ?w*
using *222* **by** *simp*
next
show *acyclic ?w*
proof (*rule kruskal-exchange-acyclic-inv-2*)
show *acyclic ?v*
using *10 spanning-forest-def* **by** *blast*
next
show *injective ?v*
using *8* **by** *simp*
next
show *?i* $\leq$*?v*
using *inf.coboundedI1* **by** *simp*
next
show *bijective* (*?i*^{T} $*$ *top*)
using *212* **by** *simp*
next
show *bijective* (*?e* $*$ *top*)
using *14 212* **by** (*smt assms*(*4*) *comp-inf.idempotent-bot-closed conv-complement minarc-arc minarc-bot-iff p-bot regular-closed-bot semiring.mult-not-zero symmetric-top-closed*)
next
show *?i* $\leq$ *top* $*$ *?e*^{T} $*$*?v*^{T*}
using *19* **by** *simp*
next
show *?v* $*$ *?e*^{T} $*$ *top* $=$ *bot*
using *13* **by** *simp*
qed
next
have *?w* $\leq$ *?v* $\sqcup$ *?e*
using *inf-le1 sup-left-isotone* **by** *simp*
also have ... $\leq$ $--g \sqcup$ *?e*
using *10 sup-left-isotone spanning-forest-def* **by** *blast*
also have ... $\leq$ $--g \sqcup --h$
proof $-$
have *1*: $--g \leq --g \sqcup --h$
by *simp*
have *2*: *?e* $\leq$ $--g \sqcup --h$
by (*metis inf.coboundedI1 inf.sup-monoid.add-commute minarc-below order.trans p-dist-inf p-dist-sup sup.cobounded1*)
thus *?thesis*
using *1 2* **by** *simp*
qed
also have ... $\leq$ $--g$
using *assms*(*18*, *19*) **by** *auto*
finally show *?w* $\leq$ $--g$

```
      by simp
    next
     have 223: ?i ≤ (?v ⊓ − ?i)^T⋆ ∗ ?e^T ∗ top
     proof (rule boruvka-exchange-spanning-inv)
      show forest ?v
       using 10 spanning-forest-def by blast
     next
      show ?v⋆ ∗ ?e^T = ?e^T
       using 13 by (smt conv-complement conv-dist-comp conv-involutive
conv-star-commute dense-pp fc-top regular-closed-top star-absorb)
     next
      show ?i ≤ ?v ⊓ top ∗ ?e^T ∗ ?w^T⋆
       using 18 inf.sup-monoid.add-assoc by auto
     next
      show arc ?i
       using 212 by blast
     next
      show arc ?e
       using assms(16) minarc-arc minarc-bot-iff by auto
     next
      show ?v ≤ −−g
       using 10 spanning-forest-def by blast
     next
      show ?w ≤ −−g
      proof −
       have 2231: ?e ≤ −−g
        by (metis inf.boundedE minarc-below pp-dist-inf)
       have ?w ≤ ?v ⊔ ?e
        using inf-le1 sup-left-isotone by simp
       also have ... ≤ −−g
        using 2231 10 spanning-forest-def sup-least by blast
       finally show ?thesis
        by blast
      qed
     next
      show ?e ≤ −−g
       by (metis inf.boundedE minarc-below pp-dist-inf)
     next
      show components g ≤ forest-components ?v
       by (simp add: 9)
     qed
     have components g ≤ forest-components ?v
      using 10 spanning-forest-def by auto
     also have ... ≤ forest-components ?w
     proof (rule kruskal-exchange-forest-components-inv)
      show injective ((?v ⊓ − ?i) ⊔ ?e)
       using 222 by simp
     next
```

```
      show regular ?i
        using 15 by simp
      next
        show ?e * top * ?e = ?e
          by (metis arc-top-arc minarc-arc minarc-bot-iff semiring.mult-not-zero)
      next
        show ?i ≤ top * ?e^T * ?v^T⋆
          using 19 by blast
      next
        show ?v * ?e^T * top = bot
          using 13 by simp
      next
        show injective ?v
          using 8 by simp
      next
        show ?i ≤ ?v
          by (simp add: le-infI1)
      next
        show ?i ≤ (?v ⊓ − ?i)^T⋆ * ?e^T * top
          using 223 by blast
      qed
      finally show components g ≤ forest-components ?w
        by simp
    next
      show regular ?w
        using 3 7 regular-conv-closed by simp
    qed
  next
    have 224: ?e ⊓ g ≠ bot
     using assms(16) inf.left-commute inf-bot-right minarc-meet-bot by fastforce
    have 225: sum (?e ⊓ g) ≤ sum (?i ⊓ g)
    proof (rule a-to-e-in-forest-modulo-equivalence)
      show symmetric g
        using assms(1) boruvka-inner-invariant-def boruvka-outer-invariant-def
by auto
    next
      show j ≠ bot
        by (simp add: assms(17))
    next
      show f ≤ −− g
        by (simp add: assms(3))
    next
      show vector j
        using assms(6) boruvka-inner-invariant-def by blast
    next
      show forest h
        by (simp add: assms(8))
    next
      show  forest-modulo-equivalence (forest-components h) d
```

by (*simp add*: *assms*(*9*))
next
show $f \sqcup f^T = h \sqcup h^T \sqcup d \sqcup d^T$
by (*simp add*: *assms*(*12*))
next
show $\forall\, a\ b.$ *forest-modulo-equivalence-path a b* (*?H*) $d \wedge a \leq - ?H \sqcap - - g \wedge b \leq d \longrightarrow sum\ (b \sqcap g) \leq sum\ (a \sqcap g)$
by (*simp add*: *assms*(*13*))
next
show *regular d*
using *assms*(*14*) **by** *auto*
next
show $?e = ?e$
by *simp*
next
show *arc ?i*
using *212* **by** *blast*
next
show *forest-modulo-equivalence-path ?i ?e ?H* $(d \sqcup ?e)$
proof −
have $d^T * ?H * ?e = bot$
using *assms*(*6*, *7*, *10* ,*11*, *17*) *dT-He-eq-bot le-bot* **by** *blast*
hence *251*: $d^T * ?H * ?e \leq (?H * d)^\star * ?H * ?e$
by *simp*
hence $d^T * ?H * ?H * ?e \leq (?H * d)^\star * ?H * ?e$
by (*metis assms*(*8*) *forest-components-star star.circ-decompose-9 mult-assoc*)
hence $d^T * (?H * d)^\star * ?H * ?e \leq (?H * d)^\star * ?H * ?e$
proof −
have $d^T * ?H * d \leq 1$
using *assms*(*9*) *forest-modulo-equivalence-def dTransHd-le-1* **by** *blast*
hence $d^T * ?H * d * (?H * d)^\star * ?H * ?e \leq (?H * d)^\star * ?H * ?e$
by (*metis mult-left-isotone star.circ-circ-mult star-involutive star-one*)
hence $d^T * ?H * ?e \sqcup d^T * ?H * d * (?H * d)^\star * ?H * ?e \leq (?H * d)^\star * ?H * ?e$
using *251* **by** *simp*
hence $d^T * (1 \sqcup ?H * d * (?H * d)^\star) * ?H * ?e \leq (?H * d)^\star * ?H * ?e$
by (*simp add*: *comp-associative comp-left-dist-sup semiring.distrib-right*)
thus *?thesis*
by (*simp add*: *star-left-unfold-equal*)
qed
hence $?H * d^T * (?H * d)^\star * ?H * ?e \leq ?H * (?H * d)^\star * ?H * ?e$
by (*simp add*: *mult-right-isotone mult-assoc*)
hence $?H * d^T * (?H * d)^\star * ?H * ?e \leq ?H * ?H * (d * ?H)^\star * ?e$
by (*smt star-slide mult-assoc*)
hence $?H * d^T * (?H * d)^\star * ?H * ?e \leq ?H * (d * ?H)^\star * ?e$
by (*metis assms*(*8*) *forest-components-star star.circ-decompose-9*)
hence $?H * d^T * (?H * d)^\star * ?H * ?e \leq (?H * d)^\star * ?H * ?e$

using *star-slide* **by** *auto*
hence $?H * d * (?H * d)^\star * ?H * ?e \sqcup ?H * d^T * (?H * d)^\star * ?H * ?e \leq (?H * d)^\star * ?H * ?e$
by (*smt le-supI star.circ-loop-fixpoint sup.cobounded2 sup-commute mult-assoc*)
hence $(?H * (d \sqcup d^T)) * (?H * d)^\star * ?H * ?e \leq (?H * d)^\star * ?H * ?e$
by (*simp add*: *semiring.distrib-left semiring.distrib-right*)
hence $(?H * (d \sqcup d^T))^\star * (?H * d)^\star * ?H * ?e \leq (?H * d)^\star * ?H * ?e$
by (*simp add*: *star-left-induct-mult mult-assoc*)
hence *252*: $(?H * (d \sqcup d^T))^\star * ?H * ?e \leq (?H * d)^\star * ?H * ?e$
by (*smt mult-left-dist-sup star.circ-transitive-equal star-slide star-sup-1 mult-assoc*)
have $?i \leq top * ?e^T * ?F$
by *auto*
hence $?i^T \leq ?F^T * ?e^{TT} * top^T$
by (*simp add*: *conv-dist-comp conv-dist-inf mult-assoc*)
hence $?i^T * top \leq ?F^T * ?e^{TT} * top^T * top$
using *comp-isotone* **by** *blast*
also have $... = ?F^T * ?e^{TT} * top^T$
by (*simp add*: *vector-mult-closed*)
also have $... = ?F * ?e^{TT} * top^T$
by (*simp add*: *conv-dist-comp conv-star-commute*)
also have $... = ?F * ?e * top$
by *simp*
also have $... = (?H * (d \sqcup d^T))^\star * ?H * ?e * top$
using *assms*(*2*, *8*, *12*) *F-is-H-and-d* **by** *simp*
also have $... \leq (?H * d)^\star * ?H * ?e * top$
by (*simp add*: *252 comp-isotone*)
also have $... \leq (?H * (d \sqcup ?e))^\star * ?H * ?e * top$
by (*simp add*: *comp-isotone star-isotone*)
finally have $?i^T * top \leq (?H * (d \sqcup ?e))^\star * ?H * ?e * top$
by *blast*
thus *?thesis*
using *212 assms*(*16*) *forest-modulo-equivalence-path-def minarc-arc minarc-bot-iff* **by** *blast*
qed
next
show $?i \leq - ?H \sqcap -- g$
proof −
have *forest-components h* ≤ *forest-components f*
using *assms*(*2*, *8*, *12*) *H-below-F* **by** *blast*
then have *241*: $?i \leq -?H$
using *16 assms*(*9*) *inf.order-lesseq-imp p-antitone-iff* **by** *blast*
have $?i \leq -- g$
using *10 inf.coboundedI1 spanning-forest-def* **by** *blast*
thus *?thesis*
using *241 inf-greatest* **by** *blast*
qed
next

show *regular h*
using *assms*(*18*) **by** *auto*
qed
have *?v* ⊓ *?e* ⊓ − *?i* = *bot*
using *14* **by** *simp*
hence *sum* (*?w* ⊓ *g*) = *sum* (*?v* ⊓ − *?i* ⊓ *g*) + *sum* (*?e* ⊓ *g*)
using *sum-disjoint inf-commute inf-assoc* **by** *simp*
also have ... ≤ *sum* (*?v* ⊓ − *?i* ⊓ *g*) + *sum* (*?i* ⊓ *g*)
using *224 225 sum-plus-right-isotone* **by** *simp*
also have ... = *sum* (((*?v* ⊓ − *?i*) ⊔ *?i*) ⊓ *g*)
using *sum-disjoint inf-le2 pseudo-complement* **by** *simp*
also have ... = *sum* ((*?v* ⊔ *?i*) ⊓ (− *?i* ⊔ *?i*) ⊓ *g*)
by (*simp add*: *sup-inf-distrib2*)
also have ... = *sum* ((*?v* ⊔ *?i*) ⊓ *g*)
using *15* **by** (*metis inf-top-right stone*)
also have ... = *sum* (*?v* ⊓ *g*)
by (*simp add*: *inf.sup-monoid.add-assoc*)
finally have *sum* (*?w* ⊓ *g*) ≤ *sum* (*?v* ⊓ *g*)
by *simp*
thus ∀ *u* . *spanning-forest u g* ⟶ *sum* (*?w* ⊓ *g*) ≤ *sum* (*u* ⊓ *g*)
using *2 11 minimum-spanning-forest-def* **by** *auto*
qed
next
have *?f* ≤ *f* ⊔ f^T ⊔ *?e*
by (*smt conv-dist-inf inf-le1 sup-left-isotone sup-mono inf.order-lesseq-imp*)
also have ... ≤ (*?v* ⊓ − *?i* ⊓ − $?i^T$) ⊔ ($?v^T$ ⊓ − *?i* ⊓ − $?i^T$) ⊔ *?e*
using *20 sup-left-isotone* **by** *simp*
also have ... ≤ (*?v* ⊓ − *?i*) ⊔ ($?v^T$ ⊓ − *?i* ⊓ − $?i^T$) ⊔ *?e*
by (*metis inf.cobounded1 sup-inf-distrib2*)
also have ... = *?w* ⊔ ($?v^T$ ⊓ − *?i* ⊓ − $?i^T$)
by (*simp add*: *sup-assoc sup-commute*)
also have ... ≤ *?w* ⊔ ($?v^T$ ⊓ − $?i^T$)
using *inf.sup-right-isotone inf-assoc sup-right-isotone* **by** *simp*
also have ... ≤ *?w* ⊔ $?w^T$
using *conv-complement conv-dist-inf conv-dist-sup sup-right-isotone* **by** *simp*
finally show *?f* ≤ *?w* ⊔ $?w^T$
by *simp*
qed
thus *?thesis* **by** *auto*
qed

lemma *boruvka-outer-invariant-when-e-not-bot*:
assumes *boruvka-inner-invariant j f h g d*
and *j* ≠ *bot*
and *selected-edge h j g* ≤ − *forest-components f*
and *selected-edge h j g* ≠ *bot*
shows *boruvka-outer-invariant* (*f* ⊓ − *selected-edge h j* g^T ⊓ − *path f h j g* ⊔ (*f* ⊓ − *selected-edge h j* g^T ⊓ *path f h j g*)^T ⊔ *selected-edge h j g*) *g*
proof −

let *?c* = *choose-component* (*forest-components h*) *j*
let *?p* = *path f h j g*
let *?F* = *forest-components f*
let *?H* = *forest-components h*
let *?e* = *selected-edge h j g*
let *?f'* = $f \sqcap -?e^T \sqcap -?p \sqcup (f \sqcap -?e^T \sqcap ?p)^T \sqcup ?e$
let *?d'* = $d \sqcup ?e$
let *?j'* = $j \sqcap -?c$
show *boruvka-outer-invariant ?f' g*
proof (*unfold boruvka-outer-invariant-def*, *intro conjI*)
show *symmetric g*
by (*meson assms*(*1*) *boruvka-inner-invariant-def boruvka-outer-invariant-def*)
next
show *injective ?f'*
proof (*rule kruskal-injective-inv*)
show *injective* ($f \sqcap - ?e^T$)
by (*meson assms*(*1*) *boruvka-inner-invariant-def boruvka-outer-invariant-def injective-inf-closed*)
show *covector* (*?p*)
using *covector-mult-closed* **by** *simp*
show $?p * (f \sqcap - ?e^T)^T \leq ?p$
by (*simp add*: *mult-right-isotone star.left-plus-below-circ star-plus mult-assoc*)
show $?e \leq ?p$
by (*meson mult-left-isotone order.trans star-outer-increasing top.extremum*)
show $?p * (f \sqcap - ?e^T)^T \leq - ?e$
proof −
have $?p * (f \sqcap - ?e^T)^T \leq ?p * f^T$
by (*simp add*: *conv-dist-inf mult-right-isotone*)
also have ... $\leq top * ?e * (f)^{T\star} * f^T$
using *conv-dist-inf star-isotone comp-isotone* **by** *simp*
also have ... $\leq - ?e$
using *assms*(*1*, *4*) *boruvka-inner-invariant-def boruvka-outer-invariant-def kruskal-injective-inv-2 minarc-arc minarc-bot-iff* **by** *auto*
finally show *?thesis* **.**
qed
show *injective* (*?e*)
by (*metis arc-injective coreflexive-bot-closed minarc-arc minarc-bot-iff semiring.mult-not-zero*)
show *coreflexive* ($?p^T * ?p \sqcap (f \sqcap - ?e^T)^T * (f \sqcap - ?e^T)$)
proof −
have $(?p^T * ?p \sqcap (f \sqcap - ?e^T)^T * (f \sqcap - ?e^T)) \leq ?p^T * ?p \sqcap f^T * f$
using *conv-dist-inf inf.sup-right-isotone mult-isotone* **by** *simp*
also have ... $\leq (top * ?e * f^{T\star})^T * (top * ?e * f^{T\star}) \sqcap f^T * f$
by (*metis comp-associative comp-inf.coreflexive-transitive comp-inf.mult-right-isotone comp-isotone conv-isotone inf.cobounded1 inf.idem inf.sup-monoid.add-commute star-isotone top.extremum*)
also have ... ≤ 1
using *assms*(*1*, *4*) *boruvka-inner-invariant-def boruvka-outer-invariant-def*

kruskal-injective-inv-3 minarc-arc minarc-bot-iff **by** *auto*
finally show *?thesis*
by *simp*
qed
qed
next
show *acyclic ?f'*
proof (*rule kruskal-acyclic-inv*)
show *acyclic* $(f \sqcap - ?e^T)$
proof −
have *f-intersect-below*: $(f \sqcap - ?e^T) \leq f$ **by** *simp*
have *acyclic f*
by (*meson assms*(*1*) *boruvka-inner-invariant-def boruvka-outer-invariant-def*)
thus *?thesis*
using *comp-isotone dual-order.trans star-isotone f-intersect-below* **by** *blast*
qed
next
show *covector ?p*
by (*metis comp-associative vector-top-closed*)
next
show $(f \sqcap - ?e^T \sqcap ?p)^T * (f \sqcap - ?e^T)^\star * ?e = bot$
proof −
have $?e \leq - (f^{T\star} * f^\star)$
by (*simp add*: *assms*(*3*))
hence $?e * top * ?e \leq - (f^{T\star} * f^\star)$
by (*metis arc-top-arc minarc-arc minarc-bot-iff semiring.mult-not-zero*)
hence $?e^T * top * ?e^T \leq - (f^{T\star} * f^\star)^T$
by (*metis comp-associative conv-complement conv-dist-comp conv-isotone symmetric-top-closed*)
hence $?e^T * top * ?e^T \leq - (f^{T\star} * f^\star)$
by (*simp add*: *conv-dist-comp conv-star-commute*)
hence $?e * (f^{T\star} * f^\star) * ?e \leq bot$
using *triple-schroeder-p* **by** *auto*
hence *1*: $?e * f^{T\star} * f^\star * ?e \leq bot$
using *mult-assoc* **by** *auto*
have *2*: $(f \sqcap - ?e^T)^{T\star} \leq f^{T\star}$
by (*simp add*: *conv-dist-inf star-isotone*)
have $(f \sqcap - ?e^T \sqcap ?p)^T * (f \sqcap - ?e^T)^\star * ?e \leq (f \sqcap ?p)^T * (f \sqcap - ?e^T)^\star * ?e$
by (*simp add*: *comp-isotone conv-dist-inf inf.orderI inf.sup-monoid.add-assoc*)
also have $... \leq (f \sqcap ?p)^T * f^\star * ?e$
by (*simp add*: *comp-isotone star-isotone*)
also have $... \leq (f \sqcap top * ?e * (f)^{T\star})^T * f^\star * ?e$
using *2* **by** (*metis comp-inf.comp-isotone comp-inf.coreflexive-transitive comp-isotone conv-isotone inf.idem top.extremum*)
also have $... = (f^T \sqcap (top * ?e * f^{T\star})^T) * f^\star * ?e$
by (*simp add*: *conv-dist-inf*)

also have ... $\leq$ *top* $*$ $(f^T \sqcap (top * ?e * f^{T\star})^T) * f^\star * ?e$
using *top-left-mult-increasing mult-assoc* **by** *auto*
also have ... $=$ $(top \sqcap top * ?e * f^{T\star}) * f^T * f^\star * ?e$
by (*smt covector-comp-inf-1 covector-mult-closed order.eq-iff inf.sup-monoid.add-commute vector-top-closed*)
also have ... $=$ *top* $*$ *?e* $*$ $f^{T\star} * f^T * f^\star * ?e$
by *simp*
also have ... $\leq$ *top* $*$ *?e* $*$ $f^{T\star} * f^\star * ?e$
by (*smt conv-dist-comp conv-isotone conv-star-commute mult-left-isotone mult-right-isotone star.left-plus-below-circ mult-assoc*)
also have ... $\leq$ *bot*
using *1 covector-bot-closed le-bot mult-assoc* **by** *fastforce*
finally show *?thesis*
using *le-bot* **by** *auto*
qed
next
show *?e* $*$ $(f \sqcap -\ ?e^T)^\star * ?e = bot$
proof $-$
have *1*: *?e* $\leq -$ *?F*
by (*simp add*: *assms*(*3*))
have *2*: *injective f*
by (*meson assms*(*1*) *boruvka-inner-invariant-def boruvka-outer-invariant-def*)
have *3*: *equivalence ?F*
using *2 forest-components-equivalence* **by** *simp*
hence *4*: $?e^T = ?e^T * top * ?e^T$
by (*smt arc-conv-closed arc-top-arc covector-complement-closed covector-conv-vector ex231e minarc-arc minarc-bot-iff pp-surjective regular-closed-top vector-mult-closed vector-top-closed*)
also have ... $\leq -$ *?F* **using** *1 3 conv-isotone conv-complement calculation* **by** *fastforce*
finally have *5*: *?e* $*$ *?F* $*$ *?e* $=$ *bot*
using *4* **by** (*smt triple-schroeder-p le-bot pp-total regular-closed-top vector-top-closed*)
have $(f \sqcap -\ ?e^T)^\star \leq f^\star$
by (*simp add*: *star-isotone*)
hence *?e* $*$ $(f \sqcap -\ ?e^T)^\star * ?e \leq ?e * f^\star * ?e$
using *mult-left-isotone mult-right-isotone* **by** *blast*
also have ... $\leq$ *?e* $*$ *?F* $*$ *?e*
by (*metis conv-star-commute forest-components-increasing mult-left-isotone mult-right-isotone star-involutive*)
also have *6*: ... $=$ *bot*
using *5* **by** *simp*
finally show *?thesis* **using** *6 le-bot* **by** *blast*
qed
next
show *forest-components* $(f \sqcap -?e^T) \leq -$ *?e*
proof $-$
have *1*: *?e* $\leq -$ *?F*

```
      by (simp add: assms(3))
    have f ⊓ − ?e^T ≤ f
      by simp
    hence forest-components (f ⊓ − ?e^T) ≤ ?F
      using forest-components-isotone by blast
    thus ?thesis
      using 1 order-lesseq-imp p-antitone-iff by blast
  qed
 qed
next
 show ?f' ≤ −−g
 proof −
  have 1: (f ⊓ − ?e^T ⊓ − ?p) ≤ −−g
   by (meson assms(1) boruvka-inner-invariant-def
boruvka-outer-invariant-def inf.coboundedI1)
  have 2: (f ⊓ − ?e^T ⊓ ?p)^T ≤ −−g
  proof −
   have (f ⊓ − ?e^T ⊓ ?p)^T ≤ f^T
    by (simp add: conv-isotone inf.sup-monoid.add-assoc)
   also have ... ≤ −−g
    by (metis assms(1) boruvka-inner-invariant-def
boruvka-outer-invariant-def conv-complement conv-isotone)
   finally show ?thesis
    by simp
  qed
  have 3: ?e ≤ −−g
   by (metis inf.boundedE minarc-below pp-dist-inf)
  show ?thesis using 1 2 3
   by simp
 qed
next
 show regular ?f'
  using assms(1) boruvka-inner-invariant-def boruvka-outer-invariant-def
minarc-regular regular-closed-star regular-conv-closed regular-mult-closed by auto
next
 show ∃ w. minimum-spanning-forest w g ∧ ?f' ≤ w ⊔ w^T
 proof (rule exists-a-w)
  show symmetric g
   using assms(1) boruvka-inner-invariant-def boruvka-outer-invariant-def by
auto
 next
  show forest f
   using assms(1) boruvka-inner-invariant-def boruvka-outer-invariant-def by
auto
 next
  show f ≤ −−g
   using assms(1) boruvka-inner-invariant-def boruvka-outer-invariant-def by
auto
 next
```

show *regular f*
using *assms(1) boruvka-inner-invariant-def boruvka-outer-invariant-def* **by** *auto*
next
show ($\exists\, w$. *minimum-spanning-forest* $w\ g \wedge f \leq w \sqcup w^T$)
using *assms(1) boruvka-inner-invariant-def boruvka-outer-invariant-def* **by** *auto*
next
show *vector j*
using *assms(1) boruvka-inner-invariant-def* **by** *blast*
next
show *regular j*
using *assms(1) boruvka-inner-invariant-def* **by** *blast*
next
show *forest h*
using *assms(1) boruvka-inner-invariant-def* **by** *blast*
next
show *forest-modulo-equivalence* (*forest-components h*) *d*
using *assms(1) boruvka-inner-invariant-def* **by** *blast*
next
show $d * top \leq - j$
using *assms(1) boruvka-inner-invariant-def* **by** *blast*
next
show *forest-components h* $* j = j$
using *assms(1) boruvka-inner-invariant-def* **by** *blast*
next
show $f \sqcup f^T = h \sqcup h^T \sqcup d \sqcup d^T$
using *assms(1) boruvka-inner-invariant-def* **by** *blast*
next
show ($\forall$ *a b* . *forest-modulo-equivalence-path a b* (*forest-components h*) $d \wedge a \leq -$(*forest-components h*) $\sqcap --\ g \wedge b \leq d \longrightarrow$ *sum*$(b \sqcap g) \leq$ *sum*$(a \sqcap g)$)
using *assms(1) boruvka-inner-invariant-def* **by** *blast*
next
show *regular d*
using *assms(1) boruvka-inner-invariant-def* **by** *blast*
next
show *selected-edge h j g* $\leq -$ *forest-components f*
by (*simp add*: *assms(3)*)
next
show *selected-edge h j g* $\neq$ *bot*
by (*simp add*: *assms(4)*)
next
show $j \neq$ *bot*
by (*simp add*: *assms(2)*)
next
show *regular h*
using *assms(1) boruvka-inner-invariant-def boruvka-outer-invariant-def* **by** *auto*
next

```
    show h ≤ −−g
      using H-below-regular-g assms(1) boruvka-inner-invariant-def
boruvka-outer-invariant-def by auto
   qed
  qed
qed

lemma second-inner-invariant-when-e-not-bot:
  assumes boruvka-inner-invariant j f h g d
    and j ≠ bot
    and selected-edge h j g ≤ − forest-components f
    and selected-edge h j g ≠ bot
  shows boruvka-inner-invariant
        (j ⊓ − choose-component (forest-components h) j)
        (f ⊓ − selected-edge h j g^T ⊓ − path f h j g ⊔
         (f ⊓ − selected-edge h j g^T ⊓ path f h j g)^T ⊔
         selected-edge h j g)
        h g (d ⊔ selected-edge h j g)
proof −
  let ?c = choose-component (forest-components h) j
  let ?p = path f h j g
  let ?F = forest-components f
  let ?H = forest-components h
  let ?e = selected-edge h j g
  let ?f' = f ⊓ −?e^T ⊓ −?p ⊔ (f ⊓ −?e^T ⊓ ?p)^T ⊔ ?e
  let ?d' = d ⊔ ?e
  let ?j' = j ⊓ −?c
  show boruvka-inner-invariant ?j' ?f' h g ?d'
  proof (unfold boruvka-inner-invariant-def, intro conjI)
    show 1: boruvka-outer-invariant ?f' g
      using assms(1, 2, 3, 4) boruvka-outer-invariant-when-e-not-bot by blast
  next
    show g ≠ bot
      using assms(1) boruvka-inner-invariant-def by force
  next
    show regular ?d'
      using assms(1) boruvka-inner-invariant-def minarc-regular by auto
  next
    show regular ?j'
      using assms(1) boruvka-inner-invariant-def by auto
  next
    show vector ?j'
      using assms(1, 2) boruvka-inner-invariant-def component-is-vector
vector-complement-closed vector-inf-closed by simp
  next
    show regular h
      by (meson assms(1) boruvka-inner-invariant-def)
  next
    show injective h
```

by (*meson assms*(*1*) *boruvka-inner-invariant-def*)
next
show *pd-kleene-allegory-class.acyclic h*
by (*meson assms*(*1*) *boruvka-inner-invariant-def*)
next
show *?H ∗ ?j′ = ?j′*
using *fc-j-eq-j-inv assms*(*1*) *boruvka-inner-invariant-def* **by** *blast*
next
show *forest-modulo-equivalence ?H ?d′*
using *assms*(*1*, *2*, *3*) *forest-modulo-equivalence-d-U-e boruvka-inner-invariant-def boruvka-outer-invariant-def* **by** *auto*
next
show *?d′ ∗ top ≤ −?j′*
proof −
have *31*: *?d′ ∗ top = d ∗ top ⊔ ?e ∗ top*
by (*simp add*: *mult-right-dist-sup*)
have *32*: *d ∗ top ≤ −?j′*
by (*meson assms*(*1*) *boruvka-inner-invariant-def inf.coboundedI1 p-antitone-iff*)
have *regular* (*?c ∗ − ?c*^{T})
using *assms*(*1*) *boruvka-inner-invariant-def boruvka-outer-invariant-def component-is-regular regular-conv-closed regular-mult-closed* **by** *presburger*
then have *minarc*(*?c ∗ − ?c*^{T} *⊓ g*) = *minarc*(*?c ⊓ − ?c*^{T} *⊓ g*)
by (*metis component-is-vector covector-comp-inf inf-top.left-neutral vector-conv-compl*)
also have *... ≤ −− (?c ⊓ − ?c*^{T} *⊓ g*)
using *minarc-below* **by** *blast*
also have *... ≤ −− ?c*
by (*simp add*: *inf.sup-monoid.add-assoc*)
also have *... = ?c*
using *component-is-regular* **by** *auto*
finally have *?e ≤ ?c*
by *simp*
then have *?e ∗ top ≤ ?c*
by (*metis component-is-vector mult-left-isotone*)
also have *... ≤ −j ⊔ ?c*
by *simp*
also have *... = − (j ⊓ − ?c)*
using *component-is-regular* **by** *auto*
finally have *33*: *?e ∗ top ≤ − (j ⊓ − ?c)*
by *simp*
show *?thesis*
using *31 32 33* **by** *auto*
qed
next
show *?f′ ⊔ ?f′*^{T} *= h ⊔ h*^{T} *⊔ ?d′ ⊔ ?d′*^{T}
proof −
have *?f′ ⊔ ?f′*^{T} *= f ⊓ − ?e*^{T} *⊓ − ?p ⊔ (f ⊓ − ?e*^{T} *⊓ ?p)*^{T} *⊔ ?e ⊔ (f ⊓ − ?e*^{T} *⊓ − ?p)*^{T} *⊔ (f ⊓ − ?e*^{T} *⊓ ?p) ⊔ ?e*^{T}

by (*simp add*: *conv-dist-sup sup-monoid.add-assoc*)
also have ... = $(f \sqcap - ?e^T \sqcap - ?p) \sqcup (f \sqcap - ?e^T \sqcap ?p) \sqcup (f \sqcap - ?e^T \sqcap ?p)^T \sqcup (f \sqcap - ?e^T \sqcap - ?p)^T \sqcup ?e^T \sqcup ?e$
by (*simp add*: *sup.left-commute sup-commute*)
also have ... = $f \sqcup f^T \sqcup ?e \sqcup ?e^T$
proof (*rule simplify-f*)
show *regular ?p*
using *assms*(*1*) *boruvka-inner-invariant-def boruvka-outer-invariant-def minarc-regular regular-closed-star regular-conv-closed regular-mult-closed* **by** *auto*
next
show *regular ?e*
using *minarc-regular* **by** *blast*
qed
also have ... = $h \sqcup h^T \sqcup d \sqcup d^T \sqcup ?e \sqcup ?e^T$
using *assms*(*1*) *boruvka-inner-invariant-def* **by** *auto*
finally show *?thesis*
by (*smt conv-dist-sup sup.left-commute sup-commute*)
qed
next
show $\forall\ a\ b\ .$ *forest-modulo-equivalence-path a b ?H ?d'* $\wedge\ a \leq - ?H \sqcap -- g \wedge\ b \leq ?d' \longrightarrow$ *sum* $(b \sqcap g) \leq$ *sum* $(a \sqcap g)$
proof (*intro allI*, *rule impI*, *unfold forest-modulo-equivalence-path-def*)
fix *a b*
assume *1*: $(arc\ a \wedge arc\ b \wedge a^T * top \leq (?H * ?d')^\star * ?H * b * top) \wedge a \leq - ?H \sqcap -- g \wedge b \leq ?d'$
thus *sum* $(b \sqcap g) \leq$ *sum* $(a \sqcap g)$
proof (*cases b* = *?e*)
case *b-equals-e*: *True*
thus *?thesis*
proof (*cases a* = *?e*)
case *True*
thus *?thesis*
using *b-equals-e* **by** *auto*
next
case *a-ne-e*: *False*
have *sum* $(b \sqcap g) \leq$ *sum* $(a \sqcap g)$
proof (*rule a-to-e-in-forest-modulo-equivalence*)
show *symmetric g*
using *assms*(*1*) *boruvka-inner-invariant-def boruvka-outer-invariant-def* **by** *auto*
next
show $j \neq bot$
by (*simp add*: *assms*(*2*))
next
show $f \leq -- g$
using *assms*(*1*) *boruvka-inner-invariant-def boruvka-outer-invariant-def* **by** *auto*
next
show *vector j*

```
            using assms(1) boruvka-inner-invariant-def by blast
          next
            show forest h
              using assms(1) boruvka-inner-invariant-def by blast
          next
            show forest-modulo-equivalence (forest-components h) d
              using assms(1) boruvka-inner-invariant-def by blast
          next
            show f ⊔ f^T = h ⊔ h^T ⊔ d ⊔ d^T
              using assms(1) boruvka-inner-invariant-def by blast
          next
            show ∀ a b. forest-modulo-equivalence-path a b (?H) d ∧ a ≤ - ?H ⊓ -
- g ∧ b ≤ d ⟶ sum (b ⊓ g) ≤ sum (a ⊓ g)
              using assms(1) boruvka-inner-invariant-def by blast
          next
            show regular d
              using assms(1) boruvka-inner-invariant-def by blast
          next
            show b = ?e
              using b-equals-e by simp
          next
            show arc a
              using 1 by simp
          next
            show forest-modulo-equivalence-path a b ?H ?d'
              using 1 forest-modulo-equivalence-path-def by simp
          next
            show a ≤ - ?H ⊓ -- g
              using 1 by simp
          next
            show regular h
              using assms(1) boruvka-inner-invariant-def
boruvka-outer-invariant-def by auto
          qed
          thus ?thesis
            by simp
        qed
      next
        case b-not-equal-e: False
        then have b-below-d: b ≤ d
          using 1 assms(4) different-arc-in-sup-arc minarc-arc minarc-bot-iff by
metis
        thus ?thesis
        proof (cases ?e ≤ d)
          case True
          then have forest-modulo-equivalence-path a b ?H d ∧ b ≤ d
            using 1 forest-modulo-equivalence-path-def sup.absorb1 by auto
          thus ?thesis
            using 1 assms(1) boruvka-inner-invariant-def by blast
```

next
case *e-not-less-than-d*: *False*
have *71*:*equivalence ?H*
using *assms*(*1*) *fch-equivalence boruvka-inner-invariant-def* **by** *auto*
then have *72*: *forest-modulo-equivalence-path a b ?H ?d′ ⟷ forest-modulo-equivalence-path a b ?H d ∨* (*forest-modulo-equivalence-path a ?e ?H d ∧ forest-modulo-equivalence-path ?e b ?H d*)
proof (*rule forest-modulo-equivalence-path-split-disj*)
show *arc ?e*
using *assms*(*4*) *minarc-arc minarc-bot-iff* **by** *blast*
next
show *regular a ∧ regular b ∧ regular ?e ∧ regular d ∧ regular ?H*
using *assms*(*1*) *1 boruvka-inner-invariant-def boruvka-outer-invariant-def arc-regular minarc-regular regular-closed-star regular-conv-closed regular-mult-closed* **by** *auto*
qed
thus *?thesis*
proof (*cases forest-modulo-equivalence-path a b ?H d*)
case *True*
have *forest-modulo-equivalence-path a b ?H d ∧ b ≤ d*
using *1 True forest-modulo-equivalence-path-def sup.absorb1* **by** (*metis assms*(*4*) *b-not-equal-e minarc-arc minarc-bot-iff different-arc-in-sup-arc*)
thus *?thesis*
using *1 assms*(*1*) *b-below-d boruvka-inner-invariant-def* **by** *auto*
next
case *False*
have *73*:*forest-modulo-equivalence-path a ?e ?H d ∧ forest-modulo-equivalence-path ?e b ?H d*
using *1 72 False forest-modulo-equivalence-path-def* **by** *blast*
have *74*: *?e ≤ −−g*
by (*metis inf.boundedE minarc-below pp-dist-inf*)
have *?H ≤ ?F*
using *assms*(*1*) *H-below-F boruvka-inner-invariant-def boruvka-outer-invariant-def* **by** *blast*
then have *?e ≤ − ?H*
using *assms*(*3*) *order-trans p-antitone* **by** *blast*
then have *?e ≤ − ?H ⊓ −−g*
using *74* **by** *simp*
then have *75*: *sum* (*b ⊓ g*) ≤ *sum* (*?e ⊓ g*)
using *assms*(*1*) *b-below-d 73 boruvka-inner-invariant-def* **by** *blast*
have *76*: *forest-modulo-equivalence-path a ?e ?H ?d′*
by (*meson 73 forest-modulo-equivalence-path-split-disj assms*(*1*) *forest-modulo-equivalence-path-def boruvka-inner-invariant-def boruvka-outer-invariant-def fch-equivalence arc-regular regular-closed-star regular-conv-closed regular-mult-closed*)
have *77*: *sum* (*?e ⊓ g*) ≤ *sum* (*a ⊓ g*)
proof (*rule a-to-e-in-forest-modulo-equivalence*)
show *symmetric g*
using *assms*(*1*) *boruvka-inner-invariant-def*

boruvka-outer-invariant-def **by** *auto*
next
show $j \neq bot$
by (*simp add*: *assms*(*2*))
next
show $f \leq -- g$
using *assms*(*1*) *boruvka-inner-invariant-def*
boruvka-outer-invariant-def **by** *auto*
next
show *vector j*
using *assms*(*1*) *boruvka-inner-invariant-def* **by** *blast*
next
show *forest h*
using *assms*(*1*) *boruvka-inner-invariant-def* **by** *blast*
next
show *forest-modulo-equivalence* (*forest-components h*) *d*
using *assms*(*1*) *boruvka-inner-invariant-def* **by** *blast*
next
show $f \sqcup f^T = h \sqcup h^T \sqcup d \sqcup d^T$
using *assms*(*1*) *boruvka-inner-invariant-def* **by** *blast*
next
show $\forall\, a\ b.$ *forest-modulo-equivalence-path a b* (*?H*) $d \wedge a \leq - ?H \sqcap$
$- - g \wedge b \leq d \longrightarrow sum\ (b \sqcap g) \leq sum\ (a \sqcap g)$
using *assms*(*1*) *boruvka-inner-invariant-def* **by** *blast*
next
show *regular d*
using *assms*(*1*) *boruvka-inner-invariant-def* **by** *blast*
next
show *?e* = *?e*
by *simp*
next
show *arc a*
using *1* **by** *simp*
next
show *forest-modulo-equivalence-path a ?e ?H ?d'*
by (*simp add*: *76*)
next
show $a \leq - ?H \sqcap --g$
using *1* **by** *simp*
next
show *regular h*
using *assms*(*1*) *boruvka-inner-invariant-def*
boruvka-outer-invariant-def **by** *auto*
qed
thus *?thesis*
using *75 order.trans* **by** *blast*
qed
qed
qed

```
    qed
  qed
qed

lemma second-inner-invariant-when-e-bot:
  assumes selected-edge h j g = bot
    and selected-edge h j g ≤ − forest-components f
    and boruvka-inner-invariant j f h g d
  shows boruvka-inner-invariant
     (j ⊓ − choose-component (forest-components h) j)
     (f ⊓ − selected-edge h j g^T ⊓ − path f h j g ⊔
      (f ⊓ − selected-edge h j g^T ⊓ path f h j g)^T ⊔
      selected-edge h j g)
     h g (d ⊔ selected-edge h j g)
proof −
  let ?c = choose-component (forest-components h) j
  let ?p = path f h j g
  let ?F = forest-components f
  let ?H = forest-components h
  let ?e = selected-edge h j g
  let ?f' = f ⊓ −?e^T ⊓ −?p ⊔ (f ⊓ −?e^T ⊓ ?p)^T ⊔ ?e
  let ?d' = d ⊔ ?e
  let ?j' = j ⊓ −?c
  show boruvka-inner-invariant ?j' ?f' h g ?d'
  proof (unfold boruvka-inner-invariant-def, intro conjI)
  next
    show boruvka-outer-invariant ?f' g
      using assms(1) assms(3) boruvka-inner-invariant-def by auto
  next
    show g ≠ bot
      using assms(3) boruvka-inner-invariant-def by blast
  next
    show regular ?d'
      using assms(1) assms(3) boruvka-inner-invariant-def by auto
  next
    show regular ?j'
      using assms(3) boruvka-inner-invariant-def by auto
  next
    show vector ?j'
      by (metis assms(3) boruvka-inner-invariant-def component-is-vector
vector-complement-closed vector-inf-closed)
  next
    show regular h
      using assms(3) boruvka-inner-invariant-def by blast
  next
    show injective h
      using assms(3) boruvka-inner-invariant-def by blast
  next
    show pd-kleene-allegory-class.acyclic h
```

using *assms(3) boruvka-inner-invariant-def* **by** *blast*
next
show *?H ∗ ?j′ = ?j′*
using *assms(3) fc-j-eq-j-inv boruvka-inner-invariant-def* **by** *blast*
next
show *forest-modulo-equivalence ?H ?d′*
using *assms(1) assms(3) boruvka-inner-invariant-def* **by** *auto*
next
show *?d′ ∗ top ≤ −?j′*
using *assms(1) assms(3) boruvka-inner-invariant-def* **by** (*metis order.trans p-antitone-inf sup-monoid.add-0-right*)
next
show *?f′ ⊔ ?f′^T = h ⊔ h^T ⊔ ?d′ ⊔ ?d′^T*
using *assms(1, 3) boruvka-inner-invariant-def* **by** *auto*
next
show ∀ *a b. forest-modulo-equivalence-path a b ?H ?d′ ∧ a ≤ −?H ⊓ −−g ∧ b ≤ ?d′ ⟶ sum(b ⊓ g) ≤ sum(a ⊓ g)*
using *assms(1, 3) boruvka-inner-invariant-def* **by** *auto*
qed
qed

4.5 Formalization and correctness proof

The following result shows that Borůvka's algorithm constructs a minimum spanning forest. We have the same postcondition as the proof of Kruskal's minimum spanning tree algorithm. We show only partial correctness.

theorem *boruvka-mst*:
VARS f j h c e d
{ *symmetric g* }
f := *bot*;
WHILE −(*forest-components f*) ⊓ *g* ≠ *bot*
INV { *boruvka-outer-invariant f g* }
DO
j := *top*;
h := *f*;
d := *bot*;
WHILE j ≠ *bot*
INV { *boruvka-inner-invariant j f h g d* }
DO
c := *choose-component* (*forest-components h*) *j*;
e := *minarc*(*c ∗ −c^T ⊓ g*);
IF e ≤ −(*forest-components f*) *THEN*
f := *f* ⊓ −*e^T*;
f := (*f* ⊓ −(*top ∗ e ∗ f^T⋆*)) ⊔ (*f* ⊓ *top ∗ e ∗ f^T⋆*)^T ⊔ *e*;
d := *d* ⊔ *e*
ELSE
SKIP
FI;
j := *j* ⊓ −*c*

```
      OD
    OD
  { minimum-spanning-forest f g }
proof vcg-simp
  assume 1: symmetric g
  show boruvka-outer-invariant bot g
    using 1 boruvka-outer-invariant-def kruskal-exists-minimal-spanning by auto
next
  fix f
  let ?F = forest-components f
  assume 1: boruvka-outer-invariant f g ∧ - ?F ⊓ g ≠ bot
  have 2: equivalence ?F
    using 1 boruvka-outer-invariant-def forest-components-equivalence by auto
  show boruvka-inner-invariant top f f g bot
  proof (unfold boruvka-inner-invariant-def, intro conjI)
    show boruvka-outer-invariant f g
      by (simp add: 1)
  next
    show g ≠ bot
      using 1 by auto
  next
    show surjective top
      by simp
  next
    show regular top
      by simp
  next
    show regular bot
      by auto
  next
    show regular f
      using 1 boruvka-outer-invariant-def by blast
  next
    show injective f
      using 1 boruvka-outer-invariant-def by blast
  next
    show pd-kleene-allegory-class.acyclic f
      using 1 boruvka-outer-invariant-def by blast
  next
    show forest-modulo-equivalence ?F bot
      by (simp add: 2 forest-modulo-equivalence-def)
  next
    show bot * top ≤ - top
      by simp
  next
    show times-top-class.total (?F)
      by (simp add: star.circ-right-top mult-assoc)
  next
    show f ⊔ f^T = f ⊔ f^T ⊔ bot ⊔ bot^T
```

```
    by simp
  next
    show ∀ a b. forest-modulo-equivalence-path a b ?F bot ∧ a ≤ − ?F ⊓ −− g ∧
b ≤ bot ⟶ sum (b ⊓ g) ≤ sum (a ⊓ g)
      by (metis (full-types) forest-modulo-equivalence-path-def bot-unique
mult-left-zero mult-right-zero top.extremum)
  qed
next
  fix f j h d
  let ?c = choose-component (forest-components h) j
  let ?p = path f h j g
  let ?F = forest-components f
  let ?H = forest-components h
  let ?e = selected-edge h j g
  let ?f' = f ⊓ −?e^T ⊓ −?p ⊔ (f ⊓ −?e^T ⊓ ?p)^T ⊔ ?e
  let ?d' = d ⊔ ?e
  let ?j' = j ⊓ −?c
  assume 1: boruvka-inner-invariant j f h g d ∧ j ≠ bot
  show (?e ≤ −?F ⟶ boruvka-inner-invariant ?j' ?f' h g ?d') ∧ (¬ ?e ≤ −?F
⟶ boruvka-inner-invariant ?j' f h g d)
  proof (intro conjI)
    show ?e ≤ −?F ⟶ boruvka-inner-invariant ?j' ?f' h g ?d'
    proof (cases ?e = bot)
      case True
      thus ?thesis
        using 1 second-inner-invariant-when-e-bot by simp
    next
      case False
      thus ?thesis
        using 1 second-inner-invariant-when-e-not-bot by simp
    qed
  next
    show ¬ ?e ≤ −?F ⟶ boruvka-inner-invariant ?j' f h g d
    proof (rule impI, unfold boruvka-inner-invariant-def, intro conjI)
      show boruvka-outer-invariant f g
        using 1 boruvka-inner-invariant-def by blast
    next
      show g ≠ bot
        using 1 boruvka-inner-invariant-def by blast
    next
      show vector ?j'
        using 1 boruvka-inner-invariant-def component-is-vector
vector-complement-closed vector-inf-closed by auto
    next
      show regular ?j'
        using 1 boruvka-inner-invariant-def by auto
    next
      show regular d
        using 1 boruvka-inner-invariant-def by blast
```

next
show *regular h*
using *1 boruvka-inner-invariant-def* **by** *blast*
next
show *injective h*
using *1 boruvka-inner-invariant-def* **by** *blast*
next
show *pd-kleene-allegory-class.acyclic h*
using *1 boruvka-inner-invariant-def* **by** *blast*
next
show *forest-modulo-equivalence ?H d*
using *1 boruvka-inner-invariant-def* **by** *blast*
next
show $d * top \leq - ?j'$
using *1* **by** (*meson boruvka-inner-invariant-def dual-order.trans p-antitone-inf*)
next
show $?H * ?j' = ?j'$
using *1 fc-j-eq-j-inv boruvka-inner-invariant-def* **by** *blast*
next
show $f \sqcup f^T = h \sqcup h^T \sqcup d \sqcup d^T$
using *1 boruvka-inner-invariant-def* **by** *blast*
next
show $\neg\ ?e \leq - ?F \Longrightarrow \forall\ a\ b.\ forest\text{-}modulo\text{-}equivalence\text{-}path\ a\ b\ ?H\ d \wedge a \leq - ?H \sqcap - - g \wedge b \leq d \longrightarrow sum(b \sqcap g) \leq sum(a \sqcap g)$
using *1 boruvka-inner-invariant-def* **by** *blast*
qed
qed
next
fix *f h d*
assume *boruvka-inner-invariant bot f h g d*
thus *boruvka-outer-invariant f g*
by (*meson boruvka-inner-invariant-def*)
next
fix *f*
assume *1*: *boruvka-outer-invariant f g* $\wedge$ $-$ *forest-components f* $\sqcap$ *g* $=$ *bot*
hence *2*:*spanning-forest f g*
proof (*unfold spanning-forest-def*, *intro conjI*)
show *injective f*
using *1 boruvka-outer-invariant-def* **by** *blast*
next
show *acyclic f*
using *1 boruvka-outer-invariant-def* **by** *blast*
next
show $f \leq - - g$
using *1 boruvka-outer-invariant-def* **by** *blast*
next
show *components g* $\leq$ *forest-components f*
proof $-$

```
      let ?F = forest-components f
      have -?F ⊓ g ≤ bot
        by (simp add: 1)
      hence --g ≤ bot ⊔ --?F
        using 1 shunting-p p-antitone pseudo-complement by auto
      hence --g ≤ ?F
        using 1 boruvka-outer-invariant-def pp-dist-comp pp-dist-star
regular-conv-closed by auto
      hence (--g)^⋆ ≤ ?F^⋆
        by (simp add: star-isotone)
      thus ?thesis
        using 1 boruvka-outer-invariant-def forest-components-star by auto
    qed
  next
    show regular f
      using 1 boruvka-outer-invariant-def by auto
  qed
  from 1 obtain w where 3: minimum-spanning-forest w g ∧ f ≤ w ⊔ w^T
    using boruvka-outer-invariant-def by blast
  hence w = w ⊓ --g
    by (simp add: inf.absorb1 minimum-spanning-forest-def spanning-forest-def)
  also have ... ≤ w ⊓ components g
    by (metis inf.sup-right-isotone star.circ-increasing)
  also have ... ≤ w ⊓ f^T^⋆ ∗ f^⋆
    using 2 spanning-forest-def inf.sup-right-isotone by simp
  also have ... ≤ f ⊔ f^T
  proof (rule cancel-separate-6[where z=w and y=w^T])
    show injective w
      using 3 minimum-spanning-forest-def spanning-forest-def by simp
  next
    show f^T ≤ w^T ⊔ w
      using 3 by (metis conv-dist-inf conv-dist-sup conv-involutive inf.cobounded2
inf.orderE)
  next
    show f ≤ w^T ⊔ w
      using 3 by (simp add: sup-commute)
  next
    show injective w
      using 3 minimum-spanning-forest-def spanning-forest-def by simp
  next
    show w ⊓ w^T^⋆ = bot
      using 3 by (metis acyclic-star-below-complement comp-inf.mult-right-isotone
inf-p le-bot minimum-spanning-forest-def spanning-forest-def)
  qed
  finally have 4: w ≤ f ⊔ f^T
    by simp
  have sum (f ⊓ g) = sum ((w ⊔ w^T) ⊓ (f ⊓ g))
    using 3 by (metis inf-absorb2 inf.assoc)
  also have ... = sum (w ⊓ (f ⊓ g)) + sum (w^T ⊓ (f ⊓ g))
```

```
    using 3 inf.commute acyclic-asymmetric sum-disjoint
minimum-spanning-forest-def spanning-forest-def by simp
  also have ... = sum (w ⊓ (f ⊓ g)) + sum (w ⊓ (f^T ⊓ g^T))
    by (metis conv-dist-inf conv-involutive sum-conv)
  also have ... = sum (f ⊓ (w ⊓ g)) + sum (f^T ⊓ (w ⊓ g))
  proof −
    have 51:f^T ⊓ (w ⊓ g) = f^T ⊓ (w ⊓ g^T)
      using 1 boruvka-outer-invariant-def by auto
    have 52:f ⊓ (w ⊓ g) = w ⊓ (f ⊓ g)
      by (simp add: inf.left-commute)
    thus ?thesis
      using 51 52 abel-semigroup.left-commute inf.abel-semigroup-axioms by
fastforce
  qed
  also have ... = sum ((f ⊔ f^T) ⊓ (w ⊓ g))
    using 2 acyclic-asymmetric inf.sup-monoid.add-commute sum-disjoint
spanning-forest-def by simp
  also have ... = sum (w ⊓ g)
    using 4 by (metis inf-absorb2 inf.assoc)
  finally show minimum-spanning-forest f g
    using 2 3 minimum-spanning-forest-def by simp
qed

end

end
```

References

[1] O. Borůvka. O jistém problému minimálním. *Práce moravské přírodovědecké společnosti*, 3(3):37–58, 1926.

[2] R. L. Graham and P. Hell. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, 7(1):43–57, 1985.

[3] W. Guttmann. Relation-algebraic verification of Prim's minimum spanning tree algorithm. In A. Sampaio and F. Wang, editors, *Theoretical Aspects of Computing – ICTAC 2016*, volume 9965 of *Lecture Notes in Computer Science*, pages 51–68. Springer, 2016.

[4] W. Guttmann. Aggregation algebras. *Archive of Formal Proofs*, 2018.

[5] W. Guttmann. An algebraic framework for minimum spanning tree problems. *Theoretical Computer Science*, 744:37–55, 2018.

[6] W. Guttmann. Verifying minimum spanning tree algorithms with Stone relation algebras. *Journal of Logical and Algebraic Methods in Programming*, 101:132–150, 2018.

[7] J. Nešetřil, E. Milková, and H. Nešetřilová. Otakar Borůvka on minimum spanning tree problem – Translation of both the 1926 papers, comments, history. *Discrete Mathematics*, 233(1–3):3–36, 2001.

[8] N. Robinson-O'Brien. A formal correctness proof of Borůvka's minimum spanning tree algorithm. Master's thesis, University of Canterbury, 2020. https://doi.org/10.26021/10196.