Relational Minimum Spanning Tree Algorithms

Walter Guttmann and Nicolas Robinson-O’Brien

December 8, 2020

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 An operation to select components . . . . . . . . . . . . . . . . 35
   4.3 m-k-Stone-Kleene relation algebras . . . . . . . . . . . . . . . 39
       4.3.1 Components of forests and big forests . . . . . . . . . . 41
       4.3.2 Identifying arcs . . . . . . . . . . . . . . . . . . . . . . 49
       4.3.3 Comparison of edge weights . . . . . . . . . . . . . . . 64
       4.3.4 Maintenance of algorithm invariants . . . . . . . . . . 72
   4.4 Formalization and correctness proof . . . . . . . . . . . . . . . . 102

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]. Certain lemmas in this theory are numbered for easy correlation to theorems from the thesis.

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 Aggregation-Algebras.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

2
definition minimum-spanning-forest \( f g \equiv \) spanning-forest \( f g \land (\forall u. \) spanning-forest \( u g \rightarrow \sum (f \cap g) \leq \sum (u \cap g)) \)

definition kruskal-spanning-invariant \( f g h \equiv \) symmetric \( g \land h = hT \land g \cap --h = h \land \) spanning-forest \( f (-h \cap g) \)

definition kruskal-invariant \( f g h \equiv \) kruskal-spanning-invariant \( f g h \land (\exists w. \) minimum-spanning-forest \( w g \land f \leq w \cup wT) \)

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 \( g = gT \)
    using assms by simp
  next
  show \( g \cap --g = g \)
    using inf.sup-monoid.add-commute selection-closed-id by simp
  next
  show spanning-forest bot \( (-g \cap g) \)
    using star.circ-transitive-equal spanning-forest-def by simp
qed

lemma kruskal-vc-2:
assumes kruskal-spanning-invariant \( f g h \)
and \( h \neq \) bot
shows \( (\minarc h \leq \neg\text{-forest-components } f \rightarrow \) kruskal-spanning-invariant \((f \cap --(\text{top} * \minarc h * fT*)) \cup (f \cap \text{top} * \minarc h * fT) \cup \minarc h) \ g (h \cap \neg\minarc h \cap \neg\minarc hT) \)
\land \( \text{card } \{x. \text{regular } x \land x \leq --h \land x \leq --\minarc h \land x \leq --\minarc hT \} < \text{card } \{x. \text{regular } x \land x \leq --h \} \) \land
\( (\neg\minarc h \leq \neg\text{-forest-components } f \rightarrow \) kruskal-spanning-invariant \( f g \)
\( (h \cap --\minarc h \cap --\minarc hT) \)
\land \( \text{card } \{x. \text{regular } x \land x \leq --h \land x \leq --\minarc h \land x \leq --\minarc hT \} < \text{card } \{x. \text{regular } x \land x \leq --h \} \)
proof
  let \(?e = \minarc h \)
  let \(?f = (f \cap --(\text{top} * \?e * fT*)) \cup (f \cap \text{top} * \?e * fT) \cup \?e \)
  let \(?h = h \cap --\?e \cap --\?eT \)
  let \(?F = \text{forest-components } f \)
  let \(?n1 = \text{card } \{x. \text{regular } x \land x \leq --h \} \)
  let \(?n2 = \text{card } \{x. \text{regular } x \land x \leq --h \land x \leq --\?e \land x \leq --\?eT \} \)
  have 1: regular \( f \land \text{regular } \?e \)
    by (metis assms(1) kruskal-spanning-invariant-def spanning-forest-def minarc-regular)
  hence 2: regular \( \?f \land \text{regular } \?F \land \text{regular } (\?eT) \)
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
proof (rule conjI)
have 5: injective ?f
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 kruskal-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 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 ?eT * top * ?eT = ?eT
using assms(2) by (simp add: arc-top-arc minarc-arc)
hence ?eT * top * ?eT ≤ − ?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 = ?hT
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 (hide-lams) 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 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-cone-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 \( \hat{h} = \hat{h}^T \)  
  using assms(1) by (simp add: conv-complement conv-dist-inf inf-commute inf-left-commute kruskal-spanning-invariant-def)
next
  show \( g \cap -\neg \hat{h} = \hat{h} \)  
  using 1 2 by (metis (hide-lams) assms(1) kruskal-spanning-invariant-def inf-assoc pp-dist-inf)
next
  show spanning-forest f \((-\hat{h} \cap g)\)  
  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 \( f \leq -(-\hat{h} \cap g) \)  
  using assms(1) kruskal-spanning-invariant-def spanning-forest-def by simp
also have \( ... \leq -(-\hat{h} \cap g) \)  
  using comp-inf.mult-right-isotone inf.sup-monoid.add-commute inf-left-commute p-antitone-inf pp-isotone by presburger
  finally show \( f \leq -(-\hat{h} \cap g) \)  
  by simp
next
  show components \((-\hat{h} \cap g) \leq \hat{F}\)  
  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 \( \hat{n}_2 < \hat{n}_1 \)  
  using 4 by simp
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** \(\text{kruskal-spanning} \):

\[
\begin{align*}
\text{VARS} & \quad e, f, h \\
& \quad \text{[ symmetric } g ] \\
& \quad f := \text{bot} \\
& \quad h := g \\
\text{WHILE} & \quad h \neq \text{bot} \\
& \quad \text{INV} \{ \text{kruskal-spanning-invariant } f, g, h \} \\
& \quad \text{VAR} \{ \text{card } \{ x \mid \text{regular } x \land x \leq \neg h \} \} \\
& \quad \text{DO} \quad e := \text{minarc } h \\
& \quad \quad \text{IF } e \leq \neg \text{forest-components } f \text{ THEN} \\
& \quad \quad \quad f := (f \cap \neg (\text{top } \ast e \ast f^T)) \sqcup (f \cap \text{top } \ast e \ast f^T) \sqcup e \\
& \quad \quad \text{ELSE} \\
& \quad \quad \quad \text{SKIP} \\
& \quad \quad \text{FI} \\
& \quad h := h \cap \neg e \sqcap \neg e^T \\
\text{OD} \\
\{ \text{spanning-forest } f, g \} \\
\text{apply vcg-tc-simp} \\
\text{using kruskal-vc-1 apply simp} \\
\text{using kruskal-vc-2 apply simp} \\
\text{using kruskal-spanning-invariant-def by auto}
\end{align*}
\]

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

**Lemma** \(\text{kruskal-exists-spanning} \):

\[
\begin{align*}
\text{symmetric } g & \Rightarrow \exists f . \text{spanning-forest } f, g \\
\text{using tc-extract-function kruskal-spanning by blast}
\end{align*}
\]

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

**Lemma** \(\text{kruskal-exists-minimal-spanning} \):

\[
\begin{align*}
\text{assumes symmetric } g \\
\text{shows } \exists f . \text{minimum-spanning-forest } f, g \\
\text{proof} \\
\text{let } \{ \text{spanning-forest } f, g \} \\
\text{have } \exists m \in \text{spanning-forest } f, g \\
\text{apply (rule finite-set-minimal)} \\
\text{using finite-regular spanning-forest-def apply simp} \\
\text{using asms kruskal-exists-spanning apply simp} \\
\text{using sum-linear by simp} \\
\text{thus } \text{minimum-spanning-forest-def by simp} \\
\text{qed}
\end{align*}
\]

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:**

\[
\text{VARS } e f h \\
\text{[ symmetric } g \text{ ]}
\]

\[
f := \text{bot};
\]

\[
h := g;
\]

\[
\text{WHILE } h \neq \text{bot}
\]

\[
\text{INV } \{ \text{kruskal-invariant } f g h \}
\]

\[
\text{VAR } \{ \text{card } \{ x : \text{regular } x \land x \leq -h \} \}
\]

\[
\text{DO } e := \text{minarc } h;
\]

\[
\text{IF } e \leq -\text{forest-components } f \text{ THEN}
\]

\[
f := (f \cap -(\text{top } \ast e \ast f^T)) \cup (f \cap \text{top } \ast e \ast f^T)^T \cup e
\]

\[
\text{ELSE}
\]

\[
\text{SKIP}
\]

\[
h := h \cap -e \cap -e^T
\]

\[
\text{OD}
\]

\[
\text{[ minimum-spanning-forest } f g \text{ ]}
\]

**Proof:**

\[
\text{vcg-tc-simp}
\]

**Assume:** symmetric g

**Thus:** kruskal-invariant bot g g

**Using:** kruskal-invariant-def

**By:** simp

**Next**

\[
\text{fix } f h n
\]

\[
\text{let } ?e = \text{minarc } h
\]

\[
\text{let } ?f = (f \cap -(\text{top } \ast ?e \ast f^T)) \cup (f \cap \text{top } \ast ?e \ast f^T)^T \cup ?e
\]

\[
\text{let } ?h = h \cap -?e \cap -?e^T
\]

\[
\text{let } ?F = \text{forest-components } f
\]

\[
\text{let } ?n1 = \text{card } \{ x : \text{regular } x \land x \leq -h \}
\]

\[
\text{let } ?n2 = \text{card } \{ x : \text{regular } x \land x \leq -h \land x \leq -?e \land x \leq -?e^T \}
\]

**Assume:** 1: kruskal-invariant-def

**By:** simp

**Hence:** 2: regular f ∧ regular w ∧ regular ?v

**Using:** 1 by (metis kruskal-invariant-def kruskal-spanning-invariant-def minimum-spanning-forest-def minarc-regular)

**Have:** (?e ≤ -?F → kruskal-invariant-def \(?f g ?h \land ?n2 < ?n1\)) ∧ (¬ ?e ≤ -?F → kruskal-invariant-def \(?f g ?h \land ?n2 < ?n1\))

**Proof:** (rule conjI)

**Show:** ?e ≤ -?F → kruskal-invariant-def \(?f g ?h \land ?n2 < ?n1\)

**Proof:**

**Assume:** 4: ?e ≤ -?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 $e^T \circ \top \circ e^T \leq -F$

using 4 5 conv-complement conv-isotone by fastforce

hence 6: $e \circ F \circ e = \bot$

using le-bot triple-schroeder-p by simp

show kruskal-invariant $?g \ ?h \ & \ ?n2 < ?n1$

proof (unfold kruskal-invariant-def, intro conjI)

show kruskal-spanning-invariant $?g \ ?h$

using 1 4 kruskal-vc-2 kruskal-invariant-def by simp

next

show $\exists w. \ minimum-spanning-forest w \ G \ & \ ?f \ \leq \ w \ \sup \ ?w^T$

proof

let $?p = w \ \cap \ \top \circ e \circ w^T$

let $?v = (w \ \cap - (\top \circ e \circ w^T)) \ \sqcup \ ?p^T$

have 7: \text{regular $?p$}

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

have 8: \text{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: \text{components $g \ \leq \ forest-components \ ?v$}

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 \text{injective $?v$}

using 8 by simp

next

show \text{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 $?v \ \leq \ --\ G$

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.coboundedI1 minimum-spanning-forest-def spanning-forest-def)

next

show \text{components $g \ \leq \ forest-components \ ?v$}

using 9 by simp

9
next

show regular ?v
  using 3 regular-closed-star regular-conv-closed regular-mult-closed by simp

qed

have 11: \( \text{sum (} ?v \cap g \text{)} = \text{sum (} w \cap g \text{)} \)
proof
  have \( \text{sum (} ?v \cap g \text{)} = \text{sum (} w \cap -(top * ?e * w^{T*}) \cap g \text{)} + \text{sum (} ?p^T \cap g \text{)} \)
  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 \( ... = \text{sum (} (w \cap -(top * ?e * w^{T*})) \cap g \text{)} + \text{sum (} ?p \cap g \text{)} \)
  using 1 kruskal-invariant-def kruskal-spanning-invariant-def
  sum-symmetric by simp
  also have \( ... = \text{sum (} w \cap g \text{)} \)
  by simp

finally show \( \text{thesis} \)
by simp

qed

have 12: \( ?v \cup ?v^T = w \cup w^T \)
proof
  have \( ?v \cup ?v^T = (w \cap -?p) \cup ?p^T \cup (w^T \cap -?p^T) \cup ?p \)
  using conv-complement conv-dist-inf conv-dist-sup inf-import-p sup-assoc by simp
  also have \( ... = w \cup w^T \)
  using 3 7 maddux-3-11-pp by simp
finally show \( \text{thesis} \)
by simp

qed

have 13: \( ?v \ast ?e^T = bot \)
apply (rule kruskal-reroot-edge)
using 1 apply (simp add: minarc-arc)
using 2 minimum-spanning-forest-def spanning-forest-def by simp
have \( ?v \cap ?e \leq ?v \cap top \ast ?e \)
using inf-sup-right-isotone top-left-mult-increasing by simp
also have \( ... \leq ?v \ast (top \ast ?e)^T \)
using covector-restrict-comp-comp covector-mult-closed vector-top-closed by simp
finally have 14: \( ?v \cap ?e = bot \)
using 13 by (metis conv-distr-comp mult-assoc le-bot mult-left-zero)
let \( ?d = ?e \cap top \ast ?e^T \ast ?e^{T*} \cap \text{top} \ast ?e \ast -?F \)
let \( ?w = (?v \cap -?d) \cap ?e \)
have 15: regular \( ?d \)
using 3 regular-closed-star regular-cone-closed regular-mult-closed by simp
have 16: \( ?F \leq \neg ?d \)
apply (rule kruskal-edge-between-components-1)
using 5 apply simp
using 1 conv-dist-comp minarc-arc mult-assoc by simp
have 17: \( f \sqcup f^T \leq (\neg v \land \neg d \sqcap \neg d^T) \sqcup (\neg v^T \land \neg d \sqcap \neg d^T) \)
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

have 17: \( f \sqcup f^T \leq (\neg v \land \neg d \sqcap \neg d^T) \sqcup (\neg v^T \land \neg d \sqcap \neg d^T) \)
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

show minimum-spanning-forest \(?w, g\) \(\land \neg f \sqcup w^T \)
proof (intro conjI)

have 18: \( ?e^T \leq ?e^* \)
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 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 \( (\neg v \land \neg d) \land ?e^T \leq ?v \land ?e^T \)
using inf-le1 mult-left-isotone by simp
hence \( (\neg v \land \neg d) \land ?e^T = \bot \)
using 13 le-bot by simp
hence 20: \( ?v \land \neg d \land ?e^* \)
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 \( ?w \leq ?v \sqcup ?e \)
using inf.le1 sup-left-isotone by simp
also have \( \ldots \leq \neg\neg g \sqcup \neg\neg h \)
by (simp add: le-supI2 minarc-below)
also have \( \ldots = \neg\neg g \)
using 1 by (metis kruskal-invariant-def
kruskal-spanning-invariant-def pp-isotone-inf sup.orderE)
finally show \( ?w \leq \neg\neg g \)
by simp
next
have 22: \( ?d \leq (?v \sqcap \neg\neg d)^T * ?e^T * \top \)
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 \( g \leq \) forest-components \( ?v \)
using 10 spanning-forest-def by auto
also have \( \ldots \leq \) 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 \( g \leq \) forest-components \( ?w \)
by simp
next
show regular \( ?w \)
using 3 7 regular-conv-closed by simp
qed

next

have 23: \( ?e \sqcap g \neq \bot \)
  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 \( g \sqcap -h \leq (g \sqcap -h)^* \)
    using star.circ-increasing by simp
  also have ... \( \leq (-(g \sqcap -h))^* \)
    using pp-increasing star-isotone by blast
  also have ... \( \leq ?F \)
    using 1 kruskal-invariant-def kruskal-spanning-invariant-def
    inf.sup-monoid.add-commute spanning-forest-def by simp
  finally have 24: \( g \sqcap -h \leq ?F \)
    by simp
  have \( ?d \leq -g \)
    using 10 inf.coboundedI1 spanning-forest-def by blast
  hence \( ?d \leq -g \sqcap -?F \)
    using 16 inf.boundedI p-antitone-iff by simp
  also have ... = \(--(g \sqcap -?F)\)
    by simp
  also have ... \( \leq -h \)
    using 24 p-shunting-swap pp-isotone by fastforce
  finally have 25: \( ?d \leq -h \)
    by simp
  have \( ?d = \bot \longrightarrow \text{top} = \bot \)
    using 19 by (metis mult-left-zero mult-right-zero)
  hence \( ?d \neq \bot \)
    using 1 le-bot by auto
  hence 26: \( ?d \sqcap h \neq \bot \)
    using 25 by (metis inf.absorb-iff2 inf-commute pseudo-complement)
  have \( \text{sum} \left(?e \sqcap g\right) = \text{sum} \left(?e \sqcap -h \sqcap g\right) \)
    by (simp add: inf.absorb1 minarc-below)
  also have ... = \(\text{sum} \left(?e \sqcap h\right)\)
    using 1 by (metis kruskal-invariant-def
      kruskal-spanning-invariant-def inf.left-commute inf.sup-monoid.add-commute)
  also have ... \( \leq \text{sum} \left(?d \sqcap h\right) \)
    using 19 26 minarc-min by simp
  also have ... = \(\text{sum} \left(?d \sqcap (-h \sqcap g)\right)\)
    using 1 kruskal-invariant-def kruskal-spanning-invariant-def
    inf-commute by simp
  also have ... = \(\text{sum} \left(?d \sqcap g\right)\)
    using 25 by (simp add: inf.absorb2 inf-assoc inf-commute)
  finally have 27: \( \text{sum} \left(?e \sqcap g\right) \leq \text{sum} \left(?d \sqcap g\right) \)
    by simp
  have \( ?v \sqcap ?e \sqcap -?d = \bot \)
    using 14 by simp
  hence \( \text{sum} \left(?w \sqcap g\right) = \text{sum} \left(?e \sqcap -?d \sqcap g\right) + \text{sum} \left(?e \sqcap g\right) \)
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 (((?e ⊓ − ?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 ⊓ − ?d) ⊔ ?d) ⊓ g)
  by (simp add: inf.sup-monoid.add-assoc)
finally have sum (?v ⊓ g) ≤ sum (u ⊓ g)
  by (simp add: inf.cobounded1 sup-inf-distrib2)
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 ... ≤ (?v ⊓ − ?d) ⊔ (?v T ⊓ − ?d) ⊕ ?e
  by (simp add: sup-assoc sup-commute)
also have ... ≤ (?w ⊓ − ?d T) ⊔ (?v T ⊓ − ?d) ⊕ ?e
  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
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 ⊓ − ?d) ⊔ (?v T ⊓ − ?d) ⊕ ?e
  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
next
show ¬ ?e ≤ − ?F → kruskal-invariant f g ?h ∧ ?n2 < ?n1
  using 1 kruskal-vc-2 kruskal-invariant-def by auto
qed
thus (?e ≤ − ?F → kruskal-invariant f g ?h ∧ ?n2 < ?n1) ∧ (¬ ?e ≤ − ?F
  → kruskal-invariant f g ?h ∧ ?n2 < ?n1)
  using 1 by blast
next
fix f h
assume 28: kruskal-invariant f g h ∧ h = 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 \( w \) \( g \) \( f \leq w \cup w^T \)
  using kruskal-invariant-def by auto
hence \( w = w \cap \neg \neg g \)
  by (simp add: inf.absorb1 minimum-spanning-forest-def spanning-forest-def)
also have \( f \leq w \cap \text{components } g \)
  by (metis inf.sup-right-isotone star_circ_increasing)
also have \( f \leq w \cap f^* \)
  using 29 spanning-forest-def inf.sup-right-isotone by simp
also have \( f \leq f^T \)
  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.cobounded inf.orderE)
using 30 apply (simp add: sup_commute)
using 30 minimum-spanning-forest-def spanning-forest-def apply simp
using 30 by (metis conv_dist inf.conv_dist_inf conv_dist_sup conv_involutive infInf_le_bot minimum-spanning-forest-def spanning-forest-def)
finally have 31: \( w \leq f \cup f^T \)
  by simp
have \( \sum (f \cap g) = \sum ((w \cup w^T) \cap (f \cap g)) \)
  using 30 by (metis inf.absorb2 inf.assoc)
also have \( \sum = \sum (w \cap (f \cap g)) + \sum (w^T \cap (f \cap g)) \)
  using 30 inf.commute acyclic_asymmetric sum_disjoint minimum_spanning_forest_def spanning_forest_def by simp
also have \( \sum = \sum (w \cap (f \cap g)) + \sum (w^T \cap (f \cap g^T)) \)
  by (metis conv_dist inf.conv_dist_inf conv_involutive sum_conv)
also have \( \sum = \sum (f \cap (w \cap g)) + \sum (f^T \cap (w \cap g)) \)
  using 28 inf.commute inf.assoc kruskal_invariant_def kruskal_spanning_invariant_def by simp
also have \( \sum = \sum ((f \cup f^T) \cap (w \cap g)) \)
  using 29 acyclic_asymmetric inf.sup_monoid add_commute sum_disjoint spanning_forest_def by simp
also have \( \sum = \sum (w \cap g) \)
  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 Aggregation-Algebras.Hoare-Logic
Aggregation-Algebras.Aggregation-Algebras

begin

context m-kleene-algebra
begin

abbreviation component g r ≡ r T * (−−g)*
definition spanning-tree t g r ≡ forest t ∧ t ≤ (component g r) T * (component g r) T ∩ −−g ∧ component g r ≤ t T * t* ∧ 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 T ∧ injective r ∧ vector r ∧ regular r
definition prim-spanning-invariant t v g r ≡ prim-precondition g r ∧ v T = r T * t* ∧ 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) T ∩ −−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-involute covector-mult-closed vector-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: antisym mult-right-isotone star-isotone spanning-tree-def)

We first show three verification conditions which are used in both cor-
rectness 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 \( r^T = r^T * \text{bot}^* \)
  by (simp add: star-absorb)
next
  let \(?ss = r * r^T \cap g\)
  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 \( \bot \leq (\text{component } ?ss r)^T * (\text{component } ?ss r) \cap -\text{--} ?ss \)
    by simp
  next
    have \( \text{component } ?ss r \leq \text{component } (r * r^T) r \)
    by (simp add: mult-right-isotone star-isotone)
    also have \( ... \leq r^T * t^* \)
    using assms by (metis inf.eq-iff p-antitone regular-one-closed star-sub-one)
    also have \( ... = r^T * \text{bot}^* \)
    by (simp add: star-circ-zero star-one)
    finally show \( \text{component } ?ss r \leq r^T * \text{bot}^* \)
  .
next
  show regular bot
  by simp
  qed
  qed

lemma prim-vc-2:
  assumes prim-spanning-invariant t v g r
  and \( v * -v^T \cap g \neq \text{bot} \)
  shows prim-spanning-invariant \( (t \cup \text{minarc } (v * -v^T \cap g)(v \cup \text{minarc } (v * -v^T \cap g)^T * \text{top}) g \cap \{ x . \text{regular } x \land x \leq \text{component } g r \land x \leq -(v \cup \text{minarc } (v * -v^T \cap g)^T * \text{top})^T \}) < \text{card } \{ x . \text{regular } x \land x \leq \text{component } g r \land x \leq -v^T \} \)
proof
  let \(?vcv = v * -v^T \cap g\)
  let \(?e = \text{minarc } ?vcv\)
  let \(?t = t \cup ?e\)
  let \(?v = v \cup ?e^T * \text{top}\)
  let \(?c = \text{component } g r\)
  let \(?g = -\text{--}g\)
  let \(?n1 = \text{card } \{ x . \text{regular } x \land x \leq ?c \land x \leq -v^T \}\)
  let \(?n2 = \text{card } \{ x . \text{regular } x \land x \leq ?c \land x \leq -v^T \}\)
  have 1: regular v \land regular (v * v^T) \land regular (?v * ?v^T) \land regular (top * ?e)
    using assms(1) by (metis prim-spanning-invariant-def spanning-tree-def)
hence 2: \( t \leq v^* v^T \cap ?g \)
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: \( ?e \leq v^* -v^T \cap ?g \)
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: \( v \)
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 cone-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 12: \( r^T \leq r^* t^* \)
by (metis mult-right-isotone order-refl semiring.mult-not-zero star.circ-separate-mult-1 star-absorb)
hence 13: \( r^T \leq v^T \)
using assms(1) by (simp add: prim-spanning-invariant-def)
have 14: \( r^T \leq v^T \)
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 \( ?t \ ?v \ ?g \ ?r \ ?n2 < ?n1 \)
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 \( ?v^T = r^T * ?t^* \)
      using assms(1) 6 7 9 by (simp add: reachable-inv prim-spanning-invariant-def prim-precondition-def spanning-tree-def)
let \( ?G = ?v \ast ?v^\top \cap ?g \)

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 \(?t \leq (\text{component } ?G r)^T \ast (\text{component } ?G r) \cap \ldots \leq \top\)

using 1 2 5 7 13 prim-subgraph-inv inf-pp-commute mst-subgraph-inv-2 by auto

next

show component \(?v \ast ?v^\top \cap ?g\) \(\leq \top \ast \top\)

proof

have 15: \(r^T \ast (v \ast v^T \cap ?g)^* \leq r^T \ast ?t^*\)

using assms(1) 1 by (metis prim-spanning-invariant-def spanning-tree-def inf-pp-commute)

have component \(?v \ast ?v^\top \cap ?g\) \(r = r^T \ast (v \ast v^T \cap ?g)^*\)

using 1 by simp

also have \(\ldots \leq r^T \ast ?t^*\)

using 2 6 7 11 13 14 15 by (metis span-inv)

finally show \(?t \text{thesis}\).

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: \(\top \ast \top \leq \top\)

proof

have top \(\ast \top = \top \ast \top^T \ast \top\)

using 11 by (metis arc-top-edge mult-assoc)

also have \(\ldots \leq v^T \ast \top\)

using 7 8 by (metis conv-dist-comp conv-isotone mult-left-isotone symmetric-top-closed)

also have \(\ldots \leq v^T \ast \top\)

using 5 mult-right-isotone by auto

also have \(\ldots = r^T \ast t^* \ast \top\)

using 13 by simp

also have \(\ldots \leq r^T \ast \top^* \ast \top\)

using 4 by (simp add: mult-left-isotone mult-right-isotone star-isotone)

also have \(\ldots \leq \top\)

19
by (simp add: comp-associative mult-right-isotone star.right-plus-below-circ)
finally show thesis
  by simp
qed

have 17: top * ?e ≤ −vT
  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: −?vT = −vT ⊓ −(top * ?e)
  using 18 by simp

hence 20: ¬ top * ?e ≤ −?vT
  using 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

lemma prim-vc-3:
  assumes prim-spanning-invariant t v g r
  and v * −vT ⊓ g = bot
  shows spanning-tree t g r
proof -
  let ?g = −−g
  have 1: regular v ∧ regular (v * vT)
    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 * −vT ⊓ ?g = bot
    using assms(2) pp-inf-bot-iff pp-pp-inf-bot-iff by simp
  have 3: vT = rT * t* ∧ vector v
    using assms(1) by (simp add: covector-mult-closed prim-invariant-def
    prim-spanning-invariant-def vector-covector prim-precondition-def)
  have 4: t ≤ v * vT ⊓ ?g
    using assms(1) 1 by (metis prim-spanning-invariant-def inf-pp-commute
    spanning-tree-def inf.boundedE)
  have rT * (v * vT ⊓ ?g) ≤ rT * t*
    using assms(1) 1 by (metis prim-spanning-invariant-def inf-pp-commute
    spanning-tree-def)
  hence 5: component g r = vT
    using 1 2 3 4 by (metis span-post)
  have regular (v * vT)
    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 * vT ⊓ ?g
    by (metis assms(1) prim-spanning-invariant-def spanning-tree-def
    spanning-tree-def)
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:

\[
\begin{align*}
\text{VARS} & \ t \ v \ e \\
\text{[ prim-precondition g r ]} \\
\text{t} & := \bot; \\
\text{v} & := r; \\
\text{WHILE} & \ v \ast -v^T \cap g \neq \bot \\
& \text{INV} \{ \text{prim-spanning-invariant t v g r} \} \\
& \text{VAR} \{ \text{card} \{ x . \text{regular } x \land x \leq \text{component g r } \cap -v^T \} \} \\
& \text{DO} \ e := \text{minarc} (v \ast -v^T \cap g); \\
& \quad t := t \sqcup e; \\
& \quad v := v \sqcup e^T \ast \text{top} \\
\text{OD} \\
\text{[ spanning-tree t g r ]} \\
\text{apply} \ \text{vcg-tc-simp} \\
\text{apply} \ (\text{simp add: prim-vc-1}) \\
\text{using} \ \text{prim-vc-2} \ \text{apply} \ \text{blast} \\
\text{using} \ \text{prim-vc-3} \ \text{by} \ \text{auto}
\end{align*}
\]

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

**lemma** prim-exists-spanning:

\[
\text{prim-precondition g r } \Rightarrow \exists t . \text{spanning-tree t g r}
\]

\[
\text{using} \ \text{tc-extract-function prim-spanning by blast}
\]

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

**lemma** prim-exists-minimal-spanning:

\[
\text{assumes} \ \text{prim-precondition g r} \\
\text{shows} \ \exists t . \text{minimum-spanning-tree t g r}
\]

\[
\text{proof} \\
\text{let } \mathcal{S} = \{ t . \text{spanning-tree t g r} \} \\
\text{have } \exists m \in \mathcal{S} . \forall z \in \mathcal{S} . \text{sum (m } \cap g) \leq \text{sum (z } \cap g) \\
\text{apply (rule finite-set-minimal)} \\
\text{using} \ \text{finite-regular spanning-tree-def apply simp} \\
\text{using} \ \text{assms prim-exists-spanning apply simp} \\
\text{using} \ \text{sum-linear by simp}
\]

21
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.

definition theorem prim:
VARS t v e
[ prim-precondition g r ∧ (∃ w . minimum-spanning-tree w g r) ]
t := bot;
v := r;
WHILE v * −vT ∩ g ≠ bot
INV { card { x . regular x ∧ x ≤ component g r ∩ −vT } }
VAR e := minarc (v * −vT ∩ g);
t := t ∪ e;
v := v ∪ eT * 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 n
let ?vcv = v * −vT ∩ g
let ?vw = v * vT ∩ g
let ?c = minarc ?vcv
let ?t = t ∪ ?c
let ?v = v ∪ ?eT * top
let ?c = component g r
let ?g = −−g
let ?n1 = card { x . regular x ∧ x ≤ ?c ∧ x ≤ −vT }
let ?n2 = card { x . regular x ∧ x ≤ ?c ∧ x ≤ −?vT }
assume 1: prim-invariant t v g r ∧ ?vcv ≠ bot ∧ ?n1 = n
hence 2: regular v ∧ regular (v * vT)
by (metis (no-types, hide-lams) prim-invariant-def
prim-spanning-invariant-def spanning-tree-def prim-precondition-def
regular-cone-closed regular-closed-star regular-mult-closed conv-involutive)
have 3: t ≤ v * vT ∩ ?g
using 1 2 by (metis (no-types, hide-lams) prim-invariant-def
prim-spanning-invariant-def spanning-tree-def inf-pp-commute inf.boundedE)
hence 4: t ≤ v * vT
by simp
have 5: t ≤ ?g
using 3 by simp
have 6: ?e ≤ v * −vT ∩ ?g
using 2 by (metis minarc-below pp-dist-inf regular-mult-closed regular-closed-p)

hence 7: ?e ≤ v * -vT
  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 ∧ t ≤ w
  by (metis prim-invariant-def)

hence 11: vector r ∧ injective r ∧ vT = rT * t* ∧ forest w ∧ t ≤ w ∧ w ≤ ?eT * ?e ∩ ?g ∧ rT * (?eT * ?e ∩ ?g)* ≤ vT * w*

  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 ≤ v

  using predecessors-reachable reachable-restrict by auto

have 13: g = gT

  using 1 prim-invariant-def prim-spanning-invariant-def prim-precondition-def

by simp

hence 14: ?gT = ?g

  using conv-complement by simp

have prim-invariant ?t ?v g r ∧ ?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 ∃ w . minimum-spanning-tree w g r ∧ ?t ≤ w

proof

  let ?f = w ∩ v * -vT ∩ top * ?e * wT*
  let ?p = w ∩ -v * -vT ∩ top * ?e * wT*
  let ?fp = w ∩ -vT ∩ top * ?e * wT*
  let ?w = (w ∩ -?fp) ∪ ?pT ∪ ?e

  have 15: regular ?f ∧ regular ?fp ∧ regular ?w


  show minimum-spanning-tree ?w g r ∧ ?t ≤ ?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 \(?w \leq \gamma \) using 10 by (simp add: le-infI1 minimum-spanning-tree-def)
next
  show \(?c \leq \alpha \) using 10 minimum-spanning-tree-def spanning-tree-def
next
  show \(?f \sqcup \ ?p = \gamma \) using 2 8 epm-1 by fastforce

24
by (simp add: inf-assoc inf-commute)
hence 21: \( ?f \cap g = ?f \cap ?ev \)
using 2 by (simp add: inf-assoc inf-commute inf-left-commute)
have 22: \( ?e \cap g = \text{minarc} \ ?ev \cap ?ev \)
using 7 by (simp add: inf.absorb2 inf.assoc inf.commute)

hence 23: \( \text{sum} (\ ?e \cap g) \leq \text{sum} (\ ?f \cap g) \)
using 15 19 20 21 by (simp add: minarc-min)

have \( ?e \neq \text{bot} \)
using 20 comp-inf.semiring.mult-not-zero semiring.mult-not-zero
by blast

hence 24: \( ?e \cap g \neq \text{bot} \)
using 22 minarc-meet-bot by auto

have \( \text{sum} (\ ?w \cap g) = \text{sum} (\ ?w \cap \text{?fp} \cap g) + \text{sum} (\ ?p \cap g) + \text{sum} (\ ?e \cap g) \)
using 7 8 10 by (metis sum-disjoint-3 epm-8 epm-9 epm-10 minimum-spanning-tree-def spanning-tree-def)
also have ... = \( \text{sum} ((\ ?w \cap \text{?fp} \cap g) \cup \ ?p \cap g) + \text{sum} (\ ?e \cap g) \)
using 11 by (metis epm-8 sum-disjoint)
also have ... \leq \text{sum} ((\ ?w \cap \text{?fp} \cap g) \cup \ ?p \cap g) + \text{sum} (\ ?f \cap g)
using 23 24 by (simp add: sum-plus-right-isotone)
also have ... = \text{sum} (\ ?w \cap ?fp \cap g) + \text{sum} (\ ?p \cap g) + \text{sum} (\ ?f \cap g)
using 11 by (metis epm-8 sum-disjoint)
also have ... = \text{sum} (\ ?w \cap ?fp \cap g) + \text{sum} (\ ?p \cap g) + \text{sum} (\ ?f \cap g)
using 13 sum-symmetric by auto
also have ... = \text{sum} ((\ ?w \cap \text{?fp} \cap g) \cup ?p \cup \ ?f) \cap g)
using 2 8 by (metis sum-disjoint-3 epm-11 epm-12 epm-13)
also have ... = \text{sum} (\ ?w \cap g)
using 2 8 15 18 epm-2 by force
finally have \( \text{sum} (\ ?w \cap g) \leq \text{sum} (\ ?w \cap g) \)

thus \( \forall \ u . \ \text{spanning-tree} \ u \ g \ r \rightarrow \text{sum} (\ ?w \cap g) \leq \text{sum} (\ ?u \cap g) \)
using 10 order-lesseq-imp minimum-spanning-tree-def by auto

qed

next

show \( ?t \leq ?w \)
using 4 8 10 mst-extends-new-tree by simp

qed

next

show \( ?n2 < ?n1 \)
using 1 prim-invariant-def prim-vc-2 by auto

qed

thus prim-invariant \( ?t \ ?v \ g \ r \land \ ?n2 < n \)
using 1 by blast

next

fix \( t \ v \)
let \( ?g = - - g \)
assume 25: \( \text{prim-invariant} \ t \ v \ g \ r \land v \ast -v^T \cap g = \text{bot} \)

hence 26: \( \text{regular} \ v \)
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 ≤ v * v^T
using 26 27 by (simp add: minimum-spanning-tree-def spanning-tree-def
inf-pp-commute)

have vector r ∧ injective r ∧ 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 Borwka

imports
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

definition big-forest H d ≡
equivalence H
\[ d \leq -H \]
\[ \text{univalent } (H \ast d) \]
\[ H \sqcap d \ast d^T \leq 1 \]
\[ (H \ast d)^+ \leq -H \]

**definition** \textit{bf-between-points} \( p q H d \equiv \text{point } p \land \text{point } q \land p \leq (H \ast d)^* \ast H \ast d \)

**definition** \textit{bf-between-arcs} \( a b H d \equiv \text{arc } a \land \text{arc } b \land a^T \ast \text{top} \leq (H \ast d)^* \ast H \ast b \ast \text{top} \)

**Theorem 3**

**lemma** \textit{He-eq-He-THe-star}:  
\[ \text{assumes } \text{arc } e \land \text{equivalence } H \]
\[ \text{shows } H \ast e = H \ast e \ast (\text{top} \ast H \ast e)^* \]
**proof** –
\[ \text{let } \exists x = H \ast e \]
\[ \text{have } 1: H \ast e \leq H \ast e \ast (\text{top} \ast H \ast e)^* \]
\[ \text{using } \text{comp-isotone star.circ-reflexive by fastforce} \]
\[ \text{have } H \ast e \ast (\text{top} \ast H \ast e)^* = H \ast e \ast (\text{top} \ast e)^* \]
\[ \text{by } (\text{metis assms(2) preorder-idempotent surjective-var}) \]
**also have** ...
\[ \leq H \ast e \ast (\text{top} \ast (\text{top} \ast e)^* \ast e) \]
\[ \text{by } (\text{metis eq-refl star.circ-mult-I}) \]
**also have** ...
\[ \leq H \ast e \ast (\text{top} \ast \text{top} \ast e) \]
\[ \text{by } (\text{simp add: star.circ-left-top}) \]
**also have** ...
\[ = H \ast e \sqcup H \ast e \ast \text{top} \ast e \]
\[ \text{by } (\text{simp add: mult.semigroup-axioms semiring.distrib-left semigroup.assoc}) \]
**finally have** 2...
\[ \leq H \ast e \]
\[ \text{using } \text{assms arc-top-arc mult-assoc by auto} \]
**thus** ?thesis
\[ \text{using } 1 \ 2 \text{ by simp} \]

**qed**

**lemma** \textit{path-through-components}:
\[ \text{assumes } \text{equivalence } H \land \text{arc } e \]
\[ \text{shows } (H \ast (d \sqcup e))^* = (H \ast d)^* \sqcup (H \ast d)^* \ast H \ast e \ast (H \ast d)^* \]
**proof** –
\[ \text{have } H \ast e \ast (H \ast d)^* \ast H \ast e \leq H \ast e \ast \text{top} \ast H \ast e \]
\[ \text{by } (\text{simp add: comp-isotone}) \]
**also have** ...
\[ = H \ast e \ast \text{top} \ast e \]
\[ \text{by } (\text{metis assms(1) preorder-idempotent surjective-var mult-assoc}) \]
**also have** ...
\[ = H \ast e \]
\[ \text{using } \text{assms(2) arc-top-arc mult-assoc by auto} \]
**finally have** 1...
\[ \leq H \ast e \]
\[ \text{by } \text{simp} \]
\[ \text{have } (H \ast (d \sqcup e))^* = (H \ast d \sqcup H \ast e)^* \]
\[ \text{by } (\text{simp add: comp-left-dist-sup}) \]
**also have** ...
\[ = (H \ast d)^* \sqcup (H \ast d)^* \ast H \ast e \ast (H \ast d)^* \]
```
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 \((f \cap -e^T \cap -p) \cup (f \cap -e^T \cap p) \cup (f \cap -e^T \cap -p)^T \cup (f \cap -e^T \cap -p)^T \cup e \equiv f \cup f^T \cup e \cup e^T\)
proof
  have \((f \cap -e^T \cap -p) \cup (f \cap -e^T \cap p) \cup (f \cap -e^T \cap -p)^T \cup (f \cap -e^T \cap -p)^T \cup e \equiv (f \cap -e^T \cap -p) \cup (f \cap -e^T \cap p) \cup (f^T \cap -e \cap p^T) \cup (f^T \cap -e \cap -p^T) \cup e \equiv e^T \cup e\)
  by (simp add: conv-complement conv-dist-inf)
  also have \(\ldots \equiv\)
  \((f \cup (f \cap -e^T \cap p)) \cap (-e^T \cup (f \cap -e^T \cap p)) \cap (-p \cup (f \cap -e^T \cap p))\)
  \(\cup ((f^T \cup (f^T \cap -e \cap -p^T)) \cap (-e \cup (f^T \cap -e \cap -p^T)) \cap (p^T \cup (f^T \cap -e \cap -p^T))\)
  \(\equiv e^T \cup e\)
  by (metis sup-inf-distrib2 sup-assoc)
  also have \(\ldots \equiv\)
  \(((f \cup f) \cap (f \cap -e^T) \cap (f \cup p) \cap (-e^T \cup f) \cap (-e^T \cap -e^T) \cap (-e^T \cup -p) \cap (-p \cup f) \cap (-p \cap -e^T) \cap (-p \cap p)\)
  \(\cup ((f^T \cup f^T) \cap (f^T \cap -e) \cap (f^T \cup -p^T) \cap (-e \cup f^T) \cap (-e \cup -e) \cap (-e \cap -p^T) \cap (p^T \cup f^T) \cap (p^T \cap -e) \cap (p^T \cup -p^T))\)
  \(\equiv e^T \cup e\)
  using sup-inf-distrib1 sup-assoc inf-assoc sup-inf-distrib1 by simp
  also have \(\ldots \equiv\)
  \(((f \cup f) \cap (f \cap -e^T) \cap (f \cup p) \cap (-e^T \cup f) \cap (-e^T \cap -e^T) \cap (-e^T \cup -p) \cap (-p \cup f) \cap (-p \cap -e^T) \cap (-p \cap p)\)
  \(\cup ((f^T \cup f^T) \cap (f^T \cap -e) \cap (f^T \cup -p^T) \cap (-e \cup f^T) \cap (-e \cup -e) \cap (-e \cap -p^T) \cap (p^T \cup f^T) \cap (p^T \cap -e) \cap (p^T \cup -p^T))\)
  \(\equiv e^T \cup e\)
  by (smt abel-semigroup.commute inf.abel-semigroup-axioms inf.left-commute sup.abel-semigroup-axioms)
  also have \(\ldots \equiv\)
  \((f \cap -e^T \cap (p \cap -p)) \cup (f^T \cap -e \cap (p^T \cap -p^T)) \equiv e^T \cup e\)
  by (smt inf.sup-monoid.add-assoc inf.sup-monoid.add-commute inf-sup-absorb sup.idem)
  also have \(\ldots \equiv\)
  \((f \cap -e^T) \cap (f^T \cap -e) \equiv e^T \cup e\)
  by (metis assms(1) conv-complement inf-top-right stone)
  also have \(\ldots \equiv\)
  \((f \cap e^T) \cap (-e^T \cap e^T) \cap (f^T \cap e) \cap (-e \cap e)\)
  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 \( (f \cap - e^T \cap - p \cup (f \cap - e^T \cap p) \cup e) \)
and injective f
shows forest-components \( (f \cap - e^T \cap - p) \cup (f \cap - e^T \cap p) \cup e) = (f \cup f^T \cup e \cup e^T)^* \)
proof –
have forest-components \( (f \cap - e^T \cap - p) \cup (f \cap - e^T \cap p) \cup e) = wcc ((f
\cap - e^T \cap - p) \cup (f \cap - e^T \cap p) \cup e) \)
  by (simp add: assms(3) forest-components-wcc)
also have \( ((f \cap - e^T \cap - p) \cup (f \cap - e^T \cap p) \cup e \cup (f \cap - e^T \cap - p)^T \cup (f \cap - e^T \cap p) \cup e)^* \)
  using conv-dist-sup sup-assoc by auto
also have \( ((f \cap - e^T \cap - p) \cup (f \cap - e^T \cap p) \cup (f \cap - e^T \cap p)^T \cup (f \cap - e^T \cap p) \cup e)^* \)
  using sup-assoc sup-commute by auto
also have \( (f \cup f^T \cup e \cup e^T)^* \)
  using assms(1, 2, 3, 4) simplify-f by auto
finally show \( \text{thesis} \)
  by simp
qed

lemma components-disj-increasing:
assumes regular p
and regular e
and injective \( (f \cap - e^T \cap - p \cup (f \cap - e^T \cap p) \cup e) \)
and injective f
shows forest-components \( f \leq forest-components (f \cap - e^T \cap - p \cup (f \cap - e^T \cap p) \cup (f \cap - e^T \cap - p)^T \cup e)^* \)
proof –
have 1: forest-components \( (f \cap - e^T \cap - p) \cup (f \cap - e^T \cap p) \cup e) = (f \cup f^T \cup e \cup e^T)^* \)
  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 \( \leq (f \cup f^T \cup e^T \cup e)^* \)
  by (simp add: le-supI2 star-isotone sup-commute)
finally show \( \text{thesis} \)
  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 big-forest-path-split-1:
assumes arc a
and equivalence H
shows \((H * d)^* * H * a * top = (H * (d \cap - a))^* * H * a * top\)

proof —
let ?H = H
let ?x = ?H * (d \cap - a)
let ?y = ?H * a
let ?a = ?H * a * top
let ?d = ?H * d
have 1: ?d* * ?a \leq ?x* * ?a
  proof —
  have ?x* * ?y * ?x* * ?a \leq ?x* * ?a * ?a
    by (smt mult-left-isotone star.circ-right-top top-right-mult-increasing mult-assoc)
  also have ...
    by (metis ex231e mult-assoc)
  finally have 11: ?x* * ?y * ?x* * ?a \leq ?x* * ?a
    by simp
  proof —
  have \((d * ?H * (d \cap a) \sqcup ?H * (d \cap - a))^* * ?a\)
    by (smt inf-top-right maddux-3-11-pp)
  thus ?thesis
    using mult-left-dist-sup by auto
qEd
also have ...
  by (metis comp-inf.coreflexive-idempotent comp-isotone inf.cobounded1
inf.sup-monoid.add-commute semiring.add-mono star-isotone top.extremum)
also have ...
  by (simp add: sup-commute mult-assoc)
also have ...
  by (smt mult-right-dist-sup star.circ-sup-9 star.circ-unfold-sum mult-assoc)
also have ...
  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 ...
  by (simp add: sup-commute mult-associ)
also have ...
  by (smt mult-left-dist-sup star.circ-sup-9 star.circ-unfold-sum mult-assoc)
also have ...
  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 ...
  using assms(1, 2) He-eq-He-THe-star arc-regular mult-assoc by auto
finally have 13: (?H * d)* * ?a \leq ?x* * ?a \sqcup ?x* * ?y * ?x* * ?a
  by (simp add: mult-assoc)
have 14: ?x* * ?y * ?x* * ?a \leq ?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 antisym mult-assoc by simp
qed

lemma dTransHd-le-1:
assumes equivalence H and univalent (H * d)
shows dT * H * d ≤ 1
proof –
have dT * HT * 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 ≤ −(H * a * top)
proof –
have a * top * a^T ≤ 1
by (metis assms(2) conv-dist-comp symmetric-top-closed vector-top-closed mult-assoc)
  hence a * top * a^T * H ≤ H
  using assms(1) comp-isotone order-trans by blast
  hence a * top * top * a^T * H ≤ H
  by (simp add: vector-mult-closed)
  hence a * top * (H * a * top)^T ≤ 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

Theorem 4

lemma expand-big-forest:
assumes big-forest H d
shows (dT * H)^* * (H * d)^* = (dT * H)^* ⊔ (H * d)^*
proof –
have (H * d)^T * H * d ≤ 1
using assms big-forest-def mult-assoc by auto
hence dT * H * H * d ≤ 1
using assms big-forest-def conv-dist-comp by auto
thus thesis
by (simp add: cancel-separate-eq comp-associative)
qed

lemma big-forest-path-bot:
assumes arc a
and a ≤ d
and big-forest H d
shows \((d \sqcap -a)^T \ast (H \ast a \ast top) \leq \bot\)
proof
have 1: \(d^T \ast H \ast d \leq 1\)
  using assms(3) big-forest-def dTransHd-le-1 by blast
hence \(d \ast -1 \ast d^T \leq -H\)
  using triple-schroeder-p by force
hence \(d \ast -1 \ast d^T \leq 1 \sqcup -H\)
  by (simp add: le-supI2)
  hence \(d \ast -1 \ast d^T \leq 1 \sqcup -H\)
    by (simp add: mult-left-dist-sup)
  hence \(d \ast -1 \ast d^T \leq 1 \sqcup -H\)
    by (simp add: le-supI)
  hence \(d \leq (1 \sqcup -H) \ast (a \ast top)\)
    by (simp add: comp-associative conv-dist-comp)
  hence \(d \leq (1 \sqcup -H) \ast (a \ast top)\)
    by (simp add: le-supI)
  hence \(d \leq a \ast top \sqcup -H \ast a \ast top\)
    by (simp add: comp-associative mult-right-dist-sup)
  also have .. \(\leq a \ast top \sqcup -H \ast a \ast top\)
    using assms(1, 3) HcompaT-le-compHaT big-forest-def sup-right-isotone by auto
finally have \(d \leq a \ast top \sqcup - (H \ast a \ast top)\)
  by simp
hence \(d \sqcap -(H \ast a \ast top) \leq a \ast top\)
  using shunting-var-p by auto
hence 2:d \(\sqcap H \ast a \ast top \leq a \ast top\)
  using inf_sup-right-isotone order.trans pp-increasing by blast
have 3:d \(\sqcap H \ast a \ast top \leq top \ast a\)
proof
  have \(d \sqcap H \ast a \ast top \leq (H \ast a \ast d \ast top^T) \ast (top \sqcap (H \ast a)^T \ast d)\)
    by (metis dedekind inf-commute)
  also have .. \(= d \ast top \sqcap H \ast a \ast a^T \ast H^T \ast d\)
    by (simp add: conv-dist-comp inf-vector-comp mult-assoc)
also have \( \leq d \top \cap H \ast a \ast d^T \ast H^T \ast d \)
using assms(2) mult-right-isotone mult-left-isotone conv-isotone
\( \text{inf.sup-right-isotone by auto} \)
also have \( \ldots = d \top \cap H \ast a \ast d^T \ast H \ast d \)
using assms(3) big-forest-def by auto
also have \( \ldots \leq d \top \cap H \ast a \ast I \)
using I by (metis inf.sup-right-isotone mult-right-isotone mult-assoc)
also have \( \ldots \leq H \ast a \)
by simp
also have \( \ldots \leq \top \ast a \)
by (simp add: mult-left-isotone)
finally have \( d \cap H \ast a \ast \top \leq \top \ast a \)
by simp
thus \(?thesis\)
by simp
qed

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

lemma big-forest-path-split-2:
assumes arc a
and a \( \leq d \)
and big-forest H d
shows \( (H \ast (d \cap -a))^* \ast H \ast a \ast \top = (H \ast ((d \cap -a) \cup (d \cap -a)^T))^* \ast (H \ast a \ast \top) \)
proof –
let \(?lhs\) = \( (H \ast (d \cap -a))^* \ast H \ast a \ast \top \)
have 1: \( d^T \ast H \ast d \leq 1 \)
using assms(3) big-forest-def dTransHd-le-1 by blast
have 2: \( H \ast a \ast \top \leq ?lhs \)
by (metis le-iff-sup star.circ-loop-fixpoint star.circ-transitive-equal
star-involutive sup-commute mult-assoc)
have \( (d \cap -a)^T \ast (H \ast (d \cap -a))^* \ast (H \ast a \ast \top) = (d \cap -a)^T \ast H \ast (d \cap -a) \ast (H \ast (d \cap -a))^* \ast (H \ast a \ast \top) \)

33
proof
  have \((d \cap a)^T \ast (H \ast (d \cap a))^\ast \ast (H \ast a \ast top) = (d \cap a)^T \ast (I \cup H \ast (d \cap a) \ast (H \ast (d \cap a))^\ast \ast (H \ast a \ast top)\)
    by (simp add: star-left-unfold-equal)
  also have \(\ldots = (d \cap a)^T \ast H \ast a \ast top \cup (d \cap a)^T \ast H \ast (d \cap a) \ast (H \ast (d \cap a))^\ast \ast (H \ast a \ast top)\)
    by (smt mult-left-dist-sup star.circ-loop-fixpoint star.circ-mult-1 star-slide sup-commute mult-assoc)
  also have \(\ldots \leq bot \cup (d \cap a)^T \ast H \ast (d \cap a) \ast (H \ast (d \cap a))^\ast \ast (H \ast a \ast top)\)
    by (metis assms(1, 2, 3) big-forest-path-bot mult-assoc le-bot)
  thus \(?lhs \leq \?thesis\)
  qed

also have \(\ldots \leq d^T \ast H \ast d \ast (H \ast (d \cap a))^\ast \ast (H \ast a \ast top)\)
  using conv-isotone inf.cobounded1 mult-isotone by auto
also have \(\ldots \leq 1 \ast (H \ast (d \cap a))^\ast \ast (H \ast a \ast top)\)
  using 1 by (metis le-iff-sup mult-right-dist-sup)
finally have 3: \((d \cap a)^T \ast (H \ast (d \cap a))^\ast \ast (H \ast a \ast top) \leq \?lhs\)
  using mult-assoc by auto
hence 4: \(H \ast (d \cap a)^T \ast (H \ast (d \cap a))^\ast \ast (H \ast a \ast top) \leq \?lhs\)
proof
  have \(H \ast (d \cap a)^T \ast (H \ast (d \cap a))^\ast \ast (H \ast a \ast top) \leq H \ast (H \ast (d \cap a))^\ast \ast H \ast a \ast top\)
    using 3 mult-right-isotone mult-assoc by auto
  also have \(\ldots = H \ast H \ast ((d \cap a) \ast H)^\ast \ast H \ast a \ast top\)
    by (metis assms(3) big-forest-def star-slide mult-assoc preorder-idempotent)
  also have \(\ldots = H \ast ((d \cap a) \ast H)^\ast \ast H \ast a \ast top\)
    using assms(3) big-forest-def preorder-idempotent by fastforce
finally show \(?thesis\)
  by (metis assms(3) big-forest-def preorder-idempotent star-slide mult-assoc)
qed

have 5: \((H \ast (d \cap a) \cup H \ast (d \cap a)^T) \ast (H \ast (d \cap a))^\ast \ast H \ast a \ast top \leq \?lhs\)
proof
  have 51: \(H \ast (d \cap a) \ast (H \ast (d \cap a))^\ast \ast H \ast a \ast top \leq (H \ast (d \cap a))^\ast \ast H \ast a \ast top\)
    using star.left-plus-below-circ mult-left-isotone by simp
  have 52: \((H \ast (d \cap a) \cup H \ast (d \cap a)^T) \ast (H \ast (d \cap a))^\ast \ast H \ast a \ast top = H \ast (d \cap a) \ast (H \ast (d \cap a))^\ast \ast H \ast a \ast top \cup H \ast (d \cap a)^T \ast (H \ast (d \cap a))^\ast \ast H \ast a \ast top\)
    using mult-right-dist-sup by auto
  hence \(\ldots \leq (H \ast (d \cap a))^\ast \ast H \ast a \ast top \cup H \ast (d \cap a)^T \ast (H \ast (d \cap a))^\ast \ast H \ast a \ast top\)
    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 \((H \ast (d \cap a) \cup H \ast (d \cap a)^T)^\ast \ast H \ast a \ast top \leq \?lhs\)
proof

have \((H \ast (d \sqcap - a) \sqcup H \ast (d \sqcap - a)^T)^\ast \ast (H \ast (d \sqcap - a))^\ast \ast H \ast a \ast top\) \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: \((H \ast ((d \sqcap - a) \sqcup (d \sqcap - a)^T))^\ast \ast H \ast a \ast top \leq ?lhs\)

using mult-left-dist-sup by auto

have 7: \((H \ast (d \sqcap - a))^\ast \ast H \ast a \ast top \leq (H \ast ((d \sqcap - a) \sqcup (d \sqcap - a)^T))^\ast \ast H \ast a \ast top\)

by (simp add: mult-left-isotone semiring.distrib-left star-isotone)

thus ?thesis

using 6 7 by (simp add: mult-assoc)

qed

end

4.2 An operation to select components

We introduce the operation choose-component.

* Axiom component-in-v expresses that the result of choose-component is contained in the set of vertices, \(v\), we are selecting from, ignoring the weights.

* Axiom component-is-vector states that the result of choose-component is a vector.

* Axiom component-is-regular states that the result of choose-component is regular.

* Axiom component-is-connected states that any two vertices from the result of choose-component are connected in \(e\).

* Axiom component-single states that the result of choose-component is closed under being connected in \(e\).

* Finally, axiom component-not-bot-when-v-bot-bot expresses that the operation choose-component returns a non-empty component if the input satisfies the given criteria.

class choose-component =

fixes choose-component :: 'a ⇒ 'a ⇒ 'a

class choose-component-algebra = choose-component + stone-relation-algebra +

assumes component-in-v: choose-component e v ≤ −−v

assumes component-is-vector: vector (choose-component e v)

assumes component-is-regular: regular (choose-component e v)
\textbf{assumes} component-is-connected: choose-component e v * (choose-component e v) \leq e \\
\textbf{assumes} component-single: choose-component e v = e * choose-component e v \\
\textbf{assumes} component-not-bot-when-v-bot-bot:
\begin{itemize}
\item regular e 
\item equivalence e 
\item vector v 
\item regular v 
\item e * v = v 
\item v \neq bot \implies choose-component e v \neq bot
\end{itemize}

\textbf{Theorem 1}

Every \textit{m-kleene-algebra} is an instance of \textit{choose-component-algebra} when the \textit{choose-component} operation is defined as follows:

\textbf{context} m-kleene-algebra

\textbf{begin}

\textbf{definition} choose-component-input-condition e v \equiv
\begin{itemize}
\item regular e 
\item equivalence e 
\item vector v 
\item regular v 
\item e * v = v
\end{itemize}

\textbf{definition} m-choose-component e v \equiv
\begin{itemize}
\item if choose-component-input-condition e v then
\item e * minarc(v) * top 
\item else 
\item bot
\end{itemize}

\textbf{sublocale} m-choose-component-algebra: choose-component-algebra \textbf{where}
choose-component = m-choose-component

\textbf{proof}
\textbf{fix} e v 
\textbf{show} m-choose-component e v \leq v 
\textbf{proof} (cases choose-component-input-condition e v) 
\textbf{case} True 
\textbf{hence} m-choose-component e v = e * minarc(v) * top 
\textbf{by} (simp add: m-choose-component-def) 
\textbf{also have} ... \leq e * v * top 
\textbf{by} (simp add: comp-isotone minarc-below) 
\textbf{also have} ... = e * v * top 
\textbf{using} True choose-component-input-condition-def \textbf{by} auto 
\textbf{also have} ... = v * top 
\textbf{using} True choose-component-input-condition-def \textbf{by} auto 
\textbf{finally show} thesis 
\textbf{using} True choose-component-input-condition-def \textbf{by} auto
\textbf{next}
\textbf{case} False
hence $m$-choose-component $e \; v = \bot$
   using False $m$-choose-component-def by auto
thus $?thesis$
   by simp
qed
next

fix $e \; v$

show vector $(m$-choose-component $e \; v)$

proof
  (cases choose-component-input-condition $e \; v$
     case True
     thus $?thesis$
       by (simp add: mult-assoc $m$-choose-component-def)
  
ext
  case False
  thus $?thesis$
    by (simp add: $m$-choose-component-def)

qed

next

fix $e \; v$

show regular $(m$-choose-component $e \; v)$
   using choose-component-input-condition-def minarc-regular regular-closed-star
   regular-mult-closed $m$-choose-component-def by auto
next

fix $e \; v$

show $m$-choose-component $e \; v \; (m$-choose-component $e \; v)^T \leq e$

proof
  (cases $v = \bot$
     case True
     thus $?thesis$
       by (simp add: True minarc-bot)
  
ex
     case False
     assume $3: \; v \neq \bot$
     hence $e \; * \; minarc(v) \; * \; top \; * \; minarc(v)^T \leq e \; * \; 1$
       using $3$ minarc-arc arc-expanded comp-associative mult-right-isotone by fastforce
     hence $e \; * \; minarc(v) \; * \; top \; * \; minarc(v)^T \; * \; e \leq e \; * \; 1 \; * \; e$

37
using mult-left-isotone by auto
also have ... = e
using 1 choose-component-input-condition-def preorder-idempotent by auto
thus ?thesis
using calculation by auto
qed
thus ?thesis
by (simp add: calculation)
next
case False
thus ?thesis
by (simp add: m-choose-component-def)
qed
next
fix e v
show m-choose-component e v = e * m-choose-component e v
proof (cases choose-component-input-condition e v)
  case True
  thus ?thesis
  by (metis choose-component-input-condition-def preorder-idempotent
m-choose-component-def mult-assoc)
next
case False
thus ?thesis
by (simp add: m-choose-component-def)
qed
next
fix e v
show regular e ∧ equivalence e ∧ vector v ∧ regular v ∧ e * v = v ∧ v ≠ bot
→ m-choose-component e v ≠ bot
proof (cases choose-component-input-condition e v)
  case True
  hence m-choose-component e v ≥ minarc(v) * top
  by (metis choose-component-input-condition-def mult-1-left mult-left-isotone
m-choose-component-def)
  also have ... ≥ minarc(v)
  using calculation dual-order.trans top-right-mult-increasing by blast
  thus ?thesis
  using True bot-unique minarc-bot-iff by fastforce
next
case False
thus ?thesis
using choose-component-input-condition-def by blast
qed
qed
end
4.3 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.

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

definition boruvka-inner-invariant $j$ $f$ $h$ $g$ $d$ $\equiv$
  boruvka-outer-invariant $f$ $g$
  $\land$ $g \neq$ bot
  $\land$ vector $j$
  $\land$ regular $j$
  $\land$ boruvka-outer-invariant $h$ $g$
  $\land$ forest $h$
  $\land$ forest-components $h \leq$ forest-components $f$
  $\land$ big-forest (forest-components $h$) $d$
  $\land$ $d$ $\ast$ top $\leq -j$
  $\land$ forest-components $h \ast j = j$
  $\land$ forest-components $f = (forest-components h \ast (d \sqcup d^T)) \ast$ forest-components $h$
  $\land$ $f \sqcup f^T = h \sqcup h^T \sqcup d \sqcup d^T$
  $\land$ ($\forall$ $a$ $b$. bf-between-arcs $a$ $b$ (forest-components $h$) $d$ $\land$ $a \leq$
    (forest-components $h$) $\sqcap$ $-- g$ $\land$ $b \leq d$
    $\rightarrow$ sum($b \sqcap g$) $\leq$ sum($a \sqcap g$))
  $\land$ regular $d$

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

\textbf{proof –}

- let ?p = path f h j g
- let ?e = selected-edge h j g
- let ?F = forest-components f

\textbf{have 1:} ?e \leq - ?F by (simp add: assms)

\textbf{have } f^T \leq ?F by (metis conv-dist-comp conv-involutive conv-order conv-star-commute forest-components-increasing)

hence \textbf{ - } ?F \leq - f^T

using \textbf{p-antitone by auto}

\textbf{hence } ?e \leq - f^T

using \textbf{1 dual-order.trans by blast}

\textbf{hence } f^{TT} \leq - ?e^T

by (simp add: p-antitone-iff)

\textbf{hence } f^{TT} \leq - ?e^T

by (metis conv-complement conv-dist-inf inf.orderE inf.orderI)

\textbf{hence } f \leq - ?e^T

by \textbf{auto}

\textbf{hence } f = f \cap - ?e^T

using \textbf{inf.orderE by blast}

\textbf{hence } f \cap - ?e^T \cap - ?p \sqcup (f \cap - ?e^T \cap ?p)^T \sqcup ?e = f \cap - ?p \sqcup (f \cap ?p)^T \sqcup ?e

\textbf{by auto}

\textbf{thus } \textbf{?thesis by auto}

\textbf{qed}

\textbf{Theorem 2}

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

\textbf{lemma} et-below-j:

\textbf{assumes} vector j

\textbf{and} regular j

\textbf{and} j \neq \text{bot}

\textbf{shows} selected-edge h j g \ast top \leq j

\textbf{proof –}

- let ?e = selected-edge h j g
- let ?c = choose-component (forest-components h) j

\textbf{have } ?e \ast top \leq - - (?c \ast - ?c^T \cap g) \ast top

using \textbf{comp-isotone minarc-below by blast}

\textbf{also have } \ldots = ( - - (?c \ast - ?c^T) \cap -) \ast top

by \textbf{simp}

\textbf{also have } \ldots = (?c \ast - ?c^T \cap -) \ast top

using \textbf{component-is-regular regular-mult-closed by auto}

\textbf{also have } \ldots = (?c \cap - ?c^T \cap -) \ast top

by (metis component-is-vector conv-complement vector-complement-closed vector-covector)

\textbf{also have } \ldots \leq ?c \ast top

40
using inf.cobounded1 mult-left-isotone order-trans by blast
also have ... ≤ 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  \( ? \text{thesis} \)
by simp
qed

4.3.1 Components of forests and big forests

We prove a number of properties about big-forest and forest-components.

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

Theorem 5

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

lemma big-forest-path-split-disj:
assumes equivalence H
and arc c
and regular a \land regular b \land regular c \land regular d \land regular H
shows \( \text{bf-between-arcs} a \ b \ H \ (d \sqcup c) \iff \text{bf-between-arcs} a \ b \ H \ d \vee \)
(\( \text{bf-between-arcs} a \ c \ H \ d \land \text{bf-between-arcs} c \ b \ H \ d \))
proof
have 1: \( \text{bf-between-arcs} a \ b \ H \ (d \sqcup c) \longrightarrow \text{bf-between-arcs} a \ b \ H \ d \vee \)
(\( \text{bf-between-arcs} a \ c \ H \ d \land \text{bf-between-arcs} c \ b \ H \ d \))
proof (rule \text{impl})
assume 11: \( \text{bf-between-arcs} a \ b \ H \ (d \sqcup c) \)

hence \( a^T \ast \top \leq (H \ast (d \sqcup c))^* \ast H \ast b \ast \top \)
by (simp add: \text{bf-between-arcs-def})

also have \(... = (H \ast d)^* \sqcup (H \ast d)^* \ast H \ast c \ast (H \ast d)^* \ast H \ast b \ast \top \)
using \text{assms}(1, 2) \text{ path-through-components} by simp

also have \(... = (H \ast d)^* \ast H \ast b \ast \top \sqcup (H \ast d)^* \ast H \ast c \ast (H \ast d)^* \ast H \ast \top \)
by simp

finally have 12: \( a^T \ast \top \leq (H \ast d)^* \ast H \ast b \ast \top \sqcup (H \ast d)^* \ast H \ast c \ast (H \ast d)^* \ast H \ast \top \)

by simp

have 13: \( a^T \ast \top \leq (H \ast d)^* \ast H \ast b \ast \top \lor a^T \ast \top \leq (H \ast d)^* \ast H \ast c \ast (H \ast d)^* \ast H \ast \top \)
proof (rule \text{point-in-vector-sup})

show point \((a^T \ast \top)\)
using 11 \text{bf-between-arcs-def} \text{ mult-assoc} by auto

next
show vector \(((H \ast d)^* \ast H \ast b \ast \top)\)
using \text{vector-mult-closed} by simp

next
show regular \(((H \ast d)^* \ast H \ast b \ast \top)\)
using \text{assms}(3) \text{ pp-dist-comp pp-dist-star} by auto

next
show \( a^T \ast \top \leq (H \ast d)^* \ast H \ast b \ast \top \sqcup (H \ast d)^* \ast H \ast c \ast (H \ast d)^* \ast H \ast \top \)
using 12 by simp

qed

thus \( \text{bf-between-arcs} a \ b \ H \ d \lor (\text{bf-between-arcs} a \ c \ H \ d \land \text{bf-between-arcs} c \ b \ H \ d) \)

proof (cases \( a^T \ast \top \leq (H \ast d)^* \ast H \ast b \ast \top \))
case \text{True}
assume \( a^T \ast \top \leq (H \ast d)^* \ast H \ast b \ast \top \)

hence \( \text{bf-between-arcs} a \ b \ H \ d \)
using 11 \text{bf-between-arcs-def} by auto

thus \text{thesis}
by simp

next
case \text{False}

have 14: \( a^T \ast \top \leq (H \ast d)^* \ast H \ast c \ast (H \ast d)^* \ast H \ast b \ast \top \)
using 13 by (simp add: False)

hence 15: \( a^T \ast \top \leq (H \ast d)^* \ast H \ast c \ast \top \)
by (metis mult-right-isotone order-leseq-imp top-greatest mult-assoc)
have $c^T \ast \top \leq (H \ast d)^* \ast H \ast b \ast \top$
proof (rule ccontr)
  assume $\neg c^T \ast \top \leq (H \ast d)^* \ast H \ast b \ast \top$
  hence $c^T \ast \top \leq \neg((H \ast d)^* \ast H \ast b \ast \top)$
  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 $c \ast (H \ast d)^* \ast H \ast b \ast \top \leq \bot$
  using schroeder-3-p mult-assoc by auto
  thus False
  using 13 False sup.absorb-iff1 mult-assoc by auto
thus False
using 13 False sup.absorb-iff1 mult-assoc by auto
qed
hence $bf\text{-between-arcs } a \ c \ H \ d \ \land \ bf\text{-between-arcs } c \ b \ H \ d$
proof (rule impI)
  assume 22: $bf\text{-between-arcs } a \ c \ H \ d$\n  hence $a^T \ast \top \leq (H \ast d)^* \ast H \ast b \ast \top$
  using $bf\text{-between-arcs-def}$ by blast
  hence $a^T \ast \top \leq (H \ast (d \sqcup c))^* \ast H \ast b \ast \top$
  by (simp add: mult-left-isotone mult-right-dist-sup mult-right-isotone
  order.trans star-isotone star-slide)
  thus $bf\text{-between-arcs } a \ b \ H \ (d \sqcup c)$
  using 22 $bf\text{-between-arcs-def}$ by blast
qed
proof (rule impI)
  assume 23: $bf\text{-between-arcs } a \ c \ H \ d$\n  hence $a^T \ast \top \leq (H \ast d)^* \ast H \ast c \ast \top$
  using $bf\text{-between-arcs-def}$ by blast
  also have ... $\leq (H \ast d)^* \ast H \ast c \ast c^T \ast c \ast \top$
  by (metis ex431c comp-inf.start-circ-sup-2 mult-isotone mult-right-isotone
  mult-assoc)
  also have ... $\leq (H \ast d)^* \ast H \ast c \ast c^T \ast \top$
  by (simp add: mult-right-isotone mult-assoc)
  also have ... $\leq (H \ast d)^* \ast H \ast c \ast (H \ast d)^* \ast H \ast b \ast \top$
  using 23 mult-right-isotone mult-assoc by (simp add: $bf\text{-between-arcs-def}$)
  also have ... $\leq (H \ast d)^* \ast H \ast b \ast \top \sqcup (H \ast d)^* \ast H \ast c \ast (H \ast d)^* \ast H$
  * b * top
  by (simp add: $bf\text{-between-arcs-def}$)
  finally have $a^T \ast \top \leq (H \ast (d \sqcup c))^* \ast H \ast b \ast \top$
using assms(1, 2) path-through-components mult-right-dist-sup by simp
thus bf-between-arcs a b H (d ⊔ c)
  using 23 bf-between-arcs-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 ≤ −j
  and forest-components h * j = j
  and j ≠ bot
  shows dT * forest-components h * selected-edge h j g ≤ bot
proof
  let ?e = selected-edge h j g
  let ?H = forest-components h
  have 1: ?e * top ≤ j
    using assms(1, 2, 5) et-below-j by auto
  have dT * ?H * ?e ≤ (d * top)T * ?H * (?e * top)
    by (simp add: comp-isotone conv-isotone top-right-mult-increasing)
  also have ... ≤ (d * top)T * ?H * j
    using 1 mult-right-isotone by auto
  also have ... ≤ (−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 coref-vector-vector-comp)
  finally show ?thesis
    using coreflexive-bot-closed le-bot by blast
qed

lemma big-forest-d-U-e:
  assumes forest f
  and vector j
  and regular j
  and forest h
  and forest-components h ≤ forest-components f
  and big-forest (forest-components h) d
  and d * top ≤ −j
  and forest-components h * j = j
  and f ⨿ fT = h ⨿ hT ⨿ d ⨿ dT
  and selected-edge h j g ≤ − forest-components f
  and selected-edge h j g ≠ bot

44
and \( j \neq \text{bot} \)
shows \( \text{big-forest} \) (\( \text{forest-components} \) \( h \)) (\( d \sqcup \text{selected-edge} \) \( h \) \( j \) \( g \))

proof (unfold \( \text{big-forest-def} \), intro conjI)
let \( ?H = \text{forest-components} \) \( h \)
let \( ?F = \text{forest-components} \) \( f \)
let \( ?e = \text{selected-edge} \) \( h \) \( j \) \( g \)
let \( ?d' = d \sqcup ?e \)
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 1: \( ?d' \leq ?H \)
proof
have \( ?H \leq ?F \)
by (simp add: assms(5))
hence 11: \( ?e \leq ?H \)
using assms(10) order-lesseq-imp p-antitone by blast
have \( d \leq ?H \)
using assms(6) big-forest-def by auto
thus \( ?thesis \)
by (simp add: 11)
qed
show univalent (\( ?H \ast ?d' \))
proof
have (\( ?H \ast ?d' \))\( ^{T} \) (\( ?H \ast ?d' \)) = \( ?d'^{T} \ast ?H^{T} \ast ?H \ast ?d' \)
using conv-dist-comp mult-assoc by auto
also have \( \ldots = ?d'^{T} \ast ?H \ast ?H \ast ?d' \)
by (simp add: conv-dist-comp conv-star-commute)
also have \( \ldots = ?d'^{T} \ast ?H \ast ?d' \)
using 01 02 by (metis preorder-idempotent mult-assoc)
finally have 21: univalent (\( ?H \ast ?d' \)) \( \leftrightarrow \) \( ?e^{T} \ast ?H \ast ?H \ast ?e \sqcup ?e^{T} \ast ?H \ast ?e \sqcup d^{T} \ast ?H \ast ?d \leq 1 \)
using conv-dist-sup semiring.distrib-left semiring.distrib-right by auto
have 22: \( ?e^{T} \ast ?H \ast ?e \leq 1 \)
proof
have 221: \( ?e^{T} \ast ?H \ast ?e \leq ?e^{T} \ast \text{top} \ast ?e \)
by (simp add: mult-left-isotone mult-right-isotone)
have arc \( ?e \)
using assms(11) minarc-arc minarc-bot-iff by blast
hence \( ?e^{T} \ast \text{top} \ast ?e \leq 1 \)
using arc-expanded by blast
thus \( ?thesis \)
using 221 dual-order.trans by blast
qed
have 24: \( d^{T} \ast ?H \ast ?e \leq 1 \)
by (metis assms(2, 3, 7, 8, 12) dT-He-eq-bot coreflexive-bot-closed le-bot)
hence \((dT * ?H * ?e)^T \leq 1^T\)
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(6) dTransHd-le-1 big-forest-def by blast
thus \(?\thesis\)
using 21 22 24 25 by simp
qed

show coreflexive \((?H \sqcap ?d' \sqcap ?d'^T)\)
proof –
have coreflexive \((?H \sqcap ?d' \sqcap ?d'^T) \iff ?H \sqcap (d \sqcup ?e) \sqcap (d^T \sqcup ?e^T) \leq 1\)
by (simp add: conv-dist-sup)
also have ... \iff ?H \sqcap (d \sqcup d^T \sqcup d * ?e^T \sqcup ?e \sqcap 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 \((?H \sqcap ?d' \sqcap ?d'^T) \iff ?H \sqcap d \sqcup d^T \sqcup ?H \sqcap d * ?e^T \sqcup ?H \sqcap ?e \sqcap d^T \sqcup ?H \sqcap ?e * ?e^T \leq 1\)
by (simp add: inf-sup-distrib1)

have 31: \(?H \sqcap d \sqcup d^T \leq 1\)
using assms(6) big-forest-def by blast
have 32: \(?H \sqcap ?e \sqcap d^T \leq 1\)
proof –
have \(?e \sqcap d^T \leq ?e \sqcap top \sqcap (d \sqcup top)^T\)
by (simp add: conv-isotone multi-isotone top-right-mult-increasing)
also have ... \leq ?e \sqcap top \sqcap j^T\)
by (metis assms(7) conv-complement conv-isotone multi-right-isotone)
also have ... \leq j * - j^T\)
using assms(2, 3, 12) et-below-j multi-left-isotone by auto
also have ... \leq - ?H\)
using 03 by (metis assms(2, 3, 8) conv-complement conv-dist-comp equivalence-top-closed multi-left-isotone schroeder-3-p vector-top-closed)
finally have \(?e \sqcap d^T \leq - ?H\)
by simp
thus \(?\thesis\)
by (metis inf-sup-distrib1 p-antitone-iff p-shunting-swap
regular-one-closed)

qed

have 33: \(?H \sqcap d \sqcap ?e^T \leq 1\)
proof –
have 331: injective \(h\)
by (simp add: assms(4))
have \(?(H \sqcap ?e \sqcap d^T)^T \leq 1\)
using 32 coreflexive-cone-closed by auto
hence \(?H \sqcap (\top \sqcap ?d^T)^T \leq 1\)
using 331 conv-dist-inf forest-components-equivalence by auto

46
thus ?thesis
  using conv-dist-comp by auto
qed
have 34: ?H ∩ ?e * ?eT ≤ 1
proof –
  have 341: arc ?e ∧ arc (?eT)
    using assms(11) minarc-arc minarc-bot-iff by auto
  have ?H ∩ ?e * ?eT ≤ ?e * ?eT
    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
proof –
  have ?e ≤ − ?F
    by (simp add: assms(10))
  hence ?F ≤ − ?e
    by (simp add: p-antitone-iff)
  hence ?F * ?F ≤ − ?e
    using assms(1) fch-equivalence by fastforce
    by (metis assms(1) fch-equivalence forest-components-star
star.circ-decompose-9)
    using triple-schroeder-p by blast
    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
    by (simp add: assms(5))
    by (simp add: mult-left-isotone)
  have d ≤ f ⊔ fT
    using assms(9) 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)
also have ... ≤ ( ?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) * (?H * (d ⊔ ?e))
using assms(4, 11) forest-components-equivalence minarc-arc minarc-bot-iff path-through-components by auto
also have ... = (?H * d) * (?H * d) * (?H * d) * (?H * d) * (?H * d) * (?H * (d ⊔ ?e))
using mult-left-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 * ?e) ⊔ (?H * d) * (?H * d) * (?H * d) * (?H * ?e)
using mult-left-dist-sup mult-assoc by auto
also have ... = (?H * d) ⊔ (?H * d) * (?H * ?e) ⊔ (?H * ?e) ⊔ (?H * ?e) ⊔ (?H * ?e) ⊔ (?H * ?e)
by (simp add: mult.locale mult-assoc)
also have ... ≤ (?H * d) ⊔ (?H * d) * (?H * ?e) ⊔ (?H * ?e) ⊔ (?H * ?e) ⊔ (?H * ?e) ⊔ (?H * ?e)
by (simp add: mult.locale mult-assoc)
also have ... ≤ ?e
by (metis comp-associative comp-inf.coreflexive-idempotent comp-inf.coreflexive-transitive comp-isotone top.extremum)
using assms(11) arc-top-arc minarc-arc minarc-bot-iff by auto
finally have \(?e \ast (?H \ast d)^* \ast ?H \ast ?e \leq ?e\)
  by simp
hence \((?H \ast d)^* \ast ?H \ast ?e \ast (?H \ast d)^* \ast ?H \ast ?e \leq (?H \ast d)^* \ast ?H \ast ?e\)
  by (simp add: comp-associative comp-isotone)
thus \(?\text{thesis}\)
  using sup-right-isotone by blast
qed
also have ...
  = (?H \ast d)\sqcup (?H \ast d)^* \ast ?H \ast ?e \sqcup (?H \ast d)^* \ast ?H \ast ?e \ast (?H \ast d)^+ \ast ?H \ast d
  by (smt eq-iff sup left-commute sup orderE sup-commute)
also have ...
  = (?H \ast d)^+ \sqcup (?H \ast d)^* \ast ?H \ast ?e \sqcup (?H \ast d)^* \ast ?H \ast ?e \ast (?H \ast d)^+
  using star-circ-plus-same mult-assoc by auto
also have ...
  = (?H \ast d)^+ \sqcup (?H \ast d)^* \ast ?H \ast ?e \ast (?H \ast d)^*
  by (simp add: star-left-unfold-equal)
also have ...
  \leq \sim ?H
  using 44 assms(6) big-forest-def by auto
finally show ?thesis
  by simp
qed

4.3.2 Identifying arcs

The expression \(d \sqcap \top e^\top H \sqcap (H d^\top)^* H a^\top \top\) 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 big-forest H d
  and bf-between-arcs a e H d
  and \(H \ast d \ast (H \ast d)^* \leq \sim H\)
  and \(\sim a^\top \ast \top \leq H \ast e \ast \top\)
  and regular a
  and regular e
  and regular H
  and regular d
shows arc \(d \sqcap \top \ast e^\top \ast H \sqcap (H d^\top)^* \ast H \ast a^\top \ast \top\)
proof –
let \(?x = d \sqcap \top \ast e^\top \ast H \sqcap (H d^\top)^* \ast H \ast a^\top \ast \top\)
have 1:regular \(?x\)
  using assms(5, 6, 7, 8) regular-closed-star regular-conv-closed regular-mult-closed by auto
have 2: \(a^\top \ast \top \ast a \leq \sim 1\)
  using arc-expanded assms(2) bf-between-arcs-def by auto
have 3: \(e \ast \top \ast e^\top \leq \sim 1\)
  using assms(2) bf-between-arcs-def arc-expanded by blast
have \( 4: \) \( \top * ?x * \top = \top \)

proof –

have \( a^T * \top \leq (H * d)^* * H * e * \top \)

using assms(2) bf-between-arcs-def by blast

also have \( ... = H * e * \top \sqcup (H * d)^* * H * d * H * e * \top \)

by (metis star.circ-loop-firpoint star.circ-plus-same sup-commute mult-assoc)

finally have \( a^T * \top \leq H * e * \top \sqcup (H * d)^* * H * d * H * e * \top \)

by simp

hence \( a^T * \top \leq H * e * \top \lor a^T * \top \leq (H * d)^* * H * d * H * e * \top \)

using assms(2, 6, 7) point-in-vector-sap bf-between-arcs-def

regular-mult-closed vector-mult-closed by auto

hence \( a^T * \top \leq (H * d)^* * H * d * H * e * \top \)

using assms(4) by blast

also have \( ... = (H * d)^* * H * d * (H * e * \top \sqcap H * e * \top) \)

by (simp add: mult-assoc)

also have \( ... = (H * d)^* * H * (d \sqcap (H * e * \top)^T) * H * e * \top \)

by (metis comp-associative covector-inf-comp-3 star.circ-left-top star.circ-top)

also have \( ... = (H * d)^* * H * (d \sqcap \top * e^T * H^T) * H * e * \top \)

using conv-dist-comp mult-assoc by auto

also have \( ... = (H * d)^* * H * (d \sqcap \top * e^T * H) * H * e * \top \)

using assms(1) by (simp add: big-forest-def)

finally have \( 2: a^T * \top \leq (H * d)^* * H * (d \sqcap \top * e^T * H) * H * e * \top \)

by simp

hence \( e * \top \leq ((H * d)^* * H * (d \sqcap \top * e^T * H) * H)^T * a^T * \top \)

proof –

have bijective \( (e * \top) \land \text{bijective} \ (a^T * \top) \)

using assms(2) bf-between-arcs-def by auto

thus \( \?thesis \)

using 2 by (metis bijective-reverse mult-assoc)

qed

also have \( ... = H^T * (d \sqcap \top * e^T * H)^T * H^T * (H * d)^* * a^T * \top \)

by (simp add: conv-dist-comp mult-assoc)

also have \( ... = H * (d \sqcap \top * e^T * H)^T * H * (H * d)^* * a^T * \top \)

using assms(1) big-forest-def by auto

also have \( ... = H * (d \sqcap \top * e^T * H)^T * H * (d^T * H)^* * a^T * \top \)

using assms(1) big-forest-def conv-dist-comp conv-star-commute by auto

also have \( ... = H * (d^T \sqcap H * e * \top) * H * (d^T * H)^* * a^T * \top \)

using assms(1) conv-dist-comp big-forest-def comp-associative conv-dist-inf

by auto

also have \( ... = H * (d^T \sqcap H * e * \top) * (H * d^T)^* * H * a^T * \top \)

by (simp add: comp-associative star-slide)

also have \( ... = H * (d^T \sqcap H * e * \top) * ((H * d^T)^* * H * a^T \top \sqcap (H * d^T)^* * H * a^T * \top) \)

using mult-assoc by auto

also have \( ... = H * (d^T \sqcap H * e * \top \sqcap ((H * d^T)^* * H * a^T * \top)^T) * (H * d^T)^* * H * a^T * \top \)

by (smt comp-inf-vector covector-comp-inf vector-conv-covector vector-top-closed mult-assoc)
also have ... = H * (d^T ∩ (top * e^T * H)^T ∩ ((H * d^T)* * H * a^T * top)^T)
* (H * d^T)* * H * a^T * top
using assms(1) big-forest-def conv-dist-comp mult-assoc by auto
also have ... = H * (d ∩ top * e^T * H ∩ (H * d^T)* * H * a^T * top)^T * (H
* d^T)* * H * a^T * top
by (simp add: conv-dist-inf)
finally have 3: e * top ≤ H * ?x^T * (H * d^T)* * H * a^T * top
by auto
have ?x ≠ bot
proof (rule ccontr)
assume ¬ ?x ≠ bot
hence e * top = bot
using 3 le-bot by auto
thus False
using assms(2, 4) bf-between-arcs-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 ≤ 1
proof –
have 51: H * (d * H)* ∩ d * H * d^T ≤ 1
proof –
have 511: d * (H * d)* ≤ − H
using assms(1, 3) big-forest-def preorder-idempotent schroeder-4-p
triple-schroeder-p by fastforce
hence (d * H)* * d ≤ − H
using star-slide by auto
hence H * (d^T * H)* ≤ − d
by (simp assms(1) big-forest-def conv-dist-comp conv-star-commute
schroeder-4-p star-slide)
hence H * (d * H)^T ≤ − d^T
using 511 by (metis assms(1) big-forest-def schroeder-5-p star-slide)
hence H * (d * H)^T ≤ − (H * d^T)
by (metis assms(3) p-antitone iff1 schroeder-4-p star-slide mult-assoc
hence H * (d * H)^T ∩ H * d^T ≤ bot
by (simp add: bot-unique pseudo-complement)
hence H * d * (H * (d * H)^T ∩ H * d^T) ≤ 1
by (simp add: bot-unique)
hence 512: H * d * H * (d * H)^T ∩ H * d * H * d^T ≤ 1
using univalent-comp-left-dist-inf assms(1) big-forest-def mult-assoc by
fastforce
hence 513: H * d * H * (d * H)^T ∩ d * H * d^T ≤ 1
proof –
have d * H * d^T ≤ H * d * H * d^T
by (metis assms(1) big-forest-def conv-dist-comp conv-involutive
mult-1-right mult-left-isotone)
thus ?thesis

51
using 512 by (smt dual-order.trans p-antitone p-shunting-swap
regular-one-closed)

qed

have \(d^T \star H \star d \leq 1 \sqcup -H\)
using assms(1) big-forest-def dTransHd-le-1 le-supI1 by blast

hence \((-1 \sqcap H) \star d^T \star H \leq -d^T\)
by (metis assms(1) big-forest-def dTransHd-le-1
inf.sup-monoid.add-commute le-infI2 p-antitone-iff regular-one-closed
schroeder-3-p mult-assoc)

hence \(d \star (-1 \sqcap H) \star d^T \leq -H\)
by (metis assms(1) big-forest-def conv-dist-comp
schroeder-3-p
regular-one-closed)

hence \(H \sqcap d \star (-1 \sqcap H) \star d^T \leq 1\)
by (metis inf.coboundedI1 p-antitone-iff p-shunting-swap
regular-one-closed)

hence \(H \sqcap d \star d^T \sqcup H \star d \star (-1 \sqcap H) \star d^T \leq 1\)
using assms(1) big-forest-def le-supI1 by blast

hence \(H \sqcap (d \star 1 \star d^T \sqcup d \star (-1 \sqcap H) \star d^T) \leq 1\)
using comp-inf.semiring.distrib-left by auto

hence \(H \sqcap (d \star (1 \sqcup (-1 \sqcap H)) \star d^T) \leq 1\)
by (simp add: mult-left-dist-sup mult-right-dist-sup)

hence 514: \(H \sqcap d \star H \star d^T \leq 1\)
by (metis assms(1) big-forest-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 \(H \sqcap d \star H \star d^T \sqcup H \star d \star (d \star H)^* \sqcap d \star H \star d^T \leq 1\)
using 513 514 by simp

hence \(d \star H \star d^T \sqcap (H \sqcup H \star d \star H \star (d \star H)^*) \leq 1\)
by (simp add: comp-inf.semiring.distrib-left
inf.sup-monoid.add-commute)

hence \(d \star H \star d^T \sqcap H \star (1 \sqcup d \star H \star (d \star H)^*) \leq 1\)
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 \(?x \star top \star ?x^T = (d \sqcap top \star e^T \star H \sqcap (H \star d^T)^* \star H \star a^T \star top)^* \star top \star (d^T \sqcap H^T \star e^T \star top)^* \star a^T \star H \star (d^T \star H)^*\)
by (simp add: conv-dist-comp conv-dist-inf conv-star-commute mult-assoc)

also have ... = (d \sqcap top \star e^T \star H \sqcap (H \star d^T)^* \star H \star a^T \star top) \star top \star (d^T \sqcap H \star c \star top \sqcap a \star H \star (d \star H)^*)
using assms(1) big-forest-def by auto

also have ... = (H \star d^T)^* \star H \star a^T \star top \sqcap (d \sqcap top \star e^T \star H) \star top \star (d^T \sqcap H \star c \star top \sqcap a \star H \star (d \star H)^*)
by (metis inf-vector-comp vector-export-comp)

also have ... = (H \star d^T)^* \star H \star a^T \star top \sqcap (d \sqcap top \star e^T \star H) \star top \star top \star (d^T \sqcap H \star c \star top \sqcap top \star a \star H \star (d \star H)^*)
by (simp add: vector-mult-closed)
also have ... = (H * d^T)^* * H * a^T * top ∩ d * ((top * e^T * H)^T ∩ top) * top * (d^T ∩ H) ⊓ top \* a * H * (d * H)^* 
by (simp add: covector-comp-inf-1 covector-mult-closed)
also have ... = (H * d^T)^* * H * a^T * top ∩ d * ((top * e^T * H)^T ∩ (H * e * top)^T) * d^T
using inf.sup-monoid.add-assoc inf.sup-monoid.add-commute by auto
also have ... = (H * d^T)^* * H * a^T * top ∩ top * a * H * (d * H)^* ∩ d * ((top * e^T * H)^T ∩ (H * e * top)^T) * d^T
also have ... = (H * d^T)^* * H * a^T * top * a * H * (d * H)^* ∩ d * (H * e * top ∩ d * (H * e * top)^T) * d^T
using assms(1) big-forest-def conv-dist-comp mult-assoc by auto
also have ... = (H * d^T)^* * H * a^T * top * a * H * (d * H)^* ∩ d * (H * e * top ∩ d * (H * e * top)^T) * d^T
proof –
have (H * d^T)^* * H * a^T * top * a * H * (d * H)^* ≤ (H * d^T)^* * H * d^T
by (metis comp-associative comp-one-closed mult-left-associative
mult-semi-associative star.circ-transitive-equal)
also have ... = (H * d^T)^* * H * (d * H)^* 
using assms(1) big-forest-def mult.semigroup-axioms preorder-idempotent
semigroup.assoc by fastforce
also have ... = (H * d^T)^* * (H * d)^* * H 
by (metis star-slide mult-assoc)
finally show thesis
using inf.sup-left-isotone by auto
qed
also have ... ≤ (H * d^T)^* * (H * d)^* * H ∩ d * H * d^T
proof –
have d * H * e * top * e^T * H * d^T ≤ 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 ... ≤ d * H * d^T 
by (metis assms(1) big-forest-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)^* * (H * d)^* * H ∩ d * H * d^T 
by (metis assms(1) big-forest-def comp-associative preorder-idempotent
star-slide)
also have \( \ldots = H \ast ((d^T \ast H)^* \sqcup (H \ast d)^*) \ast H \sqcap d \ast H \ast d^T \)

by (simp add: assms(1) expand-big-forest mult.semigroup-axioms semigroup.assoc)

also have \( \ldots = (H \ast (d^T \ast H)^*) \ast H \sqcap H \ast (H \ast d)^* \ast H) \sqcap d \ast H \ast d^T \)

by (simp add: mult-left-dist-sup mult-right-dist-sup)

also have \( \ldots = (H \ast (d^T)^*) \ast H \sqcap d \ast H \ast d^T \sqcup (d \ast H)^* \sqcap d \ast H \ast d^T \)

by (smt assms(1) big-forest-def inf-sup-distrib2 mult.semigroup-axioms preorder-idempotent star-slide semigroup.assoc)

also have \( \ldots \leq (H \ast d^T)^* \ast H \sqcap d \ast H \ast d^T \sqcup 1 \)

using 51 \text{comp-inf}.semiring.add-left-mono by blast

finally have \( \{x \ast top \ast ?x^T \leq 1 \)

using 51 by (smt assms(1) big-forest-def conv-dist-comp conv-dist-inf conv-dist-sup cone-involutive cone-star-commute equivalence-one-closed mult.semigroup-axioms sup.absorb2 semigroup.assoc conv-isotone conv-order)

thus \( \exists x \)

by simp

qed

have 6: \(?x^T \ast top \ast ?x \leq 1 \)

proof –

have \( \exists x \ast top \ast ?x = (d^T \sqcap H^T \ast e^{TT} \ast top^T \sqcap top^{TT} \ast a^{TT} \ast H^T \ast (d^{TT} \ast H^T)^*) \ast top \ast (d \sqcap top \ast e^{TT} \ast H \sqcap (H \ast d^T)^*) \ast H \ast a^T \ast top) \)

by (simp add: conv-dist-comp conv-dist-inf conv-star-commute mult-assoc)

also have \( \ldots = (d^T \sqcap H \ast e \ast top \sqcap top \ast a \ast H \ast (d \ast H)^*) \ast top \ast (d \sqcap top \ast e^{TT} \ast H \sqcap (H \ast d^T)^*) \ast H \ast a^T \ast top) \)

using assms(1) big-forest-def by auto

also have \( \ldots = H \ast e \ast top \sqcap (d^T \sqcap top \ast a \ast H \ast (d \ast H)^*) \ast top \ast (d \sqcap top \ast e^{TT} \ast H \sqcap (H \ast d^T)^*) \ast H \ast a^T \ast top) \)

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 \( \ldots = H \ast e \ast top \sqcap d^T \ast ((top \ast a \ast H \ast (d \ast H)^*)^T \ast top) \ast (d \sqcap top \ast e^{TT} \ast H \sqcap (H \ast d^T)^*) \ast H \ast a^T \ast top) \)

by (simp add: coector-comp-inf-1 coector-mult-closed)

also have \( \ldots = H \ast e \ast top \sqcap d^T \ast (d \ast H)^T \ast H \ast a^T \ast top \ast (d \sqcap (H \ast d^T)^*) \ast H \ast a^T \ast top) \)

using assms(1) big-forest-def conv-associative conv-dist-comp by auto

also have \( \ldots = H \ast e \ast top \sqcap d^T \ast (d \ast H)^T \ast H \ast a^T \ast top \ast (d \sqcap (H \ast d^T)^*) \ast H \ast a^T \ast top) \)

by (smt comp-associative inf.sup-mono-id add-convcomp)

also have \( \ldots = H \ast e \ast top \sqcap d^T \ast (d \ast H)^T \ast H \ast a^T \ast (top \sqcap ((H \ast d^T)^*) \ast H \ast a^T \ast top) \ast d \sqcap top \ast e^{TT} \ast H \)

by (metis comp-associative conv-inf-vector vector-cone-convector vector-top-closed)

also have \( \ldots = H \ast e \ast top \sqcap (H \ast e \ast top)^T \sqcap d^T \ast (d \ast H)^T \ast H \ast a^T \ast ((H \ast d^T)^*) \ast H \ast a^T \ast top)^T \ast d \)

by (smt assms(1) big-forest-def conv-dist-comp inf.left-commute inf.sup-monoid.add-commute symmetric-top-closed mult-assoc inf.top.left-neutral)

also have \( \ldots = H \ast e \ast top \ast (H \ast e \ast top)^T \sqcap d^T \ast (d \ast H)^T \ast H \ast a^T \ast ((H \ast d^T)^*) \ast H \ast a^T \ast top)^T \ast d \)

54
using vector-covector vector-mult-closed by auto
also have ... = H * e * top * top^T * e^T * H \cap d^T * (d * H)^T * H * a^T * top^T * a^T * H \cap d^T * (H * d^T)^T * H * a^T * top * a * H * (d * H)^T * H * a^T * top * a
by (smt cone-dist-comp mult.semigroup-axioms symmetric-top-closed semigroup.assoc)
also have ... = H * e * top * top * e^T * H \cap d^T * (H * d^T)^T * H * a^T * top * a * H * (d * H)^T * H * d^T * (H *
using assms(1) big-forest-def conv-dist-comp conv-star-commute by auto
also have ... = H * e * top * e^T * H \cap d^T * (H * d^T)^T * H * a^T * top * a
* H * (d * H)^T * d
using vector-top-closed mult-assoc by auto
also have ... \leq H \cap d^T * (H * d^T)^T * H * (d * H)^T * 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) big-forest-def preorder-idempotent by auto
finally have 611: H * e * top * e^T * H \leq H
by simp
have d^T * (H * d^T)^T * H * a^T * top * a * H * (d * H)^T * d \leq d^T * (H *
d^T)^T * H + 1 * H * (d * H)^T * d
using 2 by (metis comp-associative mult-left-isotone mult-right-isotone)
also have ... = d^T * (H * d^T)^T * H * (d * H)^T * d
using assms(1) big-forest-def mult.semigroup-axioms preorder-idempotent semigroup.assoc by fastforce
finally have d^T * (H * d^T)^T * H * a^T * top * a * H * (d * H)^T * d \leq d^T
* (H * d^T)^T * H * (d * H)^T * d
by simp
thus ?thesis
using 611 comp-inf.comp-isotone by blast
qed
also have ... = H \cap (d^T * H)^T * d^T * H * d * (H * d)^T
proof –
have (d^T * H)^T * d^T * H * d * (H * d)^T \leq (d^T * H)^T * I * (H * d)^T
by (smt assms(1) big-forest-def conv-dist-comp mult-left-isotone
mult-right-isotone preorder-idempotent mult-assoc)
also have ... = (d^T * H)^T * (H * d)^T
by simp
finally show ?thesis
using inf.sap-right-isotone by blast
qed
also have ... = H \cap (d^T * H)^T \sqcup (H * d)^T
by (simp add: assms(1) expand-big-forest)
also have ... = H \cap (d^T * H)^T \sqcup H \cap (H * d)^T
by (simp add: comp-inf.semiring.distrib-left)
also have ... = I \sqcup H \cap (d^T * H)^T \sqcup H \cap (H * d)^T
proof –
have 612: $H \cap (H \star d)^* = 1 \sqcup H \cap (H \star d)^+$
using assms(1) big-forest-def reflexive-inf-star by blast
have $H \cap (d^T \star H)^* = 1 \sqcup H \cap (d^T \star H)^+$
using assms(1) big-forest-def reflexive-inf-star by auto
thus ?thesis
using 612 sup-assoc sup-commute by auto
qed
also have ...
\leq 1
proof –
have 613: $H \cap (H \star d)^+ \leq 1$
by (metis assms(3) inf.coboundedI1 p-antitone-iff p-shunting-swap
regular-one-closed)
hence $H \cap (d^T \star H)^+ \leq 1$
by (metis assms(1) big-forest-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 $(?x \star top)$
using 4 5 6 arc-expanded by blast
have bijective $(?x^T \star top)$
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, $i = v \cap \overline{Fe} \cap \top \cap e^T F$, 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 $(F \cap v)$
and regular $v$
and $v \star e^T = \bot$
and $e \star F \star e = \bot$
and $e^T \leq v^*$
and $e \neq \bot$
sows arc $(v \cap \overline{F}e \cap \top \cap e^T \star F)$
proof –
let $?i = v \cap \overline{F}e \cap \top \cap e^T \star F$
have 1: $?i^T \star top \cap ?i \leq 1$
proof –
have $?i^T \star top \cap ??i = (v^T \cap top \star e^T \star \overline{F} \cap F \star e \star top) \star top \star (v \cap \overline{F}$
also have ... = F * e * top ⊓ (v ⊓ top * e * −F) * top * e * (v ⊓ −F * e * top) ⊓ top * e * −F
by (smt covector-comp-inf covector-mult-closed inf-vector-comp vector-export-comp vector-top-closed)
also have ... = (v ⊓ top * e * −F) * top * e * −F
by (simp add: comp-associative)
also have ... = (v ⊓ top * e * −F) * (top * e * −F) * top * (v ⊓ −F * e * top)
using comp-associative comp-inf-vector I by auto
also have ... = (v ⊓ top * e * −F) * (top * (top * e * −F) * (top ⊓ (−F * e * top) ⊓ top * e * −F) * top * e * −F)
by (smt comp-inf-vector conv-dist-comp mult.semigroup-axioms)
also have ... = F * e * top ⊓ v ⊓ top * e * −F * e * top ⊓ top * e * −F * v ⊓ top * e * −F
by (simp add: asms(1))
also have ... = F * e * top ⊓ v ⊓ top * e * −F * e * top ⊓ top * e * −F * v ⊓ top * e * −F
by (metis comp-associative comp-inf-covector inf.sup-monoid.add-associac inf-top.left-neutral vector-top-closed)
also have ... = (v ⊓ e * top ⊓ top * e * −F * v ⊓ top ⊓ e * −F)
using asms(3) injective-comp-right-dist-inf multifunctor by auto
also have ... = (v ⊓ e * top ⊓ top * e * −F * e * (F ⊓ −F * v))
using asms(3) conv-dist-comp inf.sup-monoid.add-associac inf.sup-monoid.add-commute mult.semigroup-axioms univalent-comp-left-dist-inf semigroup-axioms by fastforce
also have ... = (F ⊓ v ⊓ e * top ⊓ top * e * −F * (F ⊓ −F * v))
by (metis comp-associative comp-inf-covector inf.top.left-neutral vector-top-closed)
also have ... = (F ⊓ v ⊓ e * top ⊓ top * e * −F)
by (simp add: comp-associative)
also have ... ≤ (F ⊓ v ⊓ e * −F) * (F ⊓ −F * v)
by (smt asms(3) conv-dist-comp mult-left-isotone shunt-bijective symmetric-top-closed top-right-mult-increasing mult-associac)
also have ... ≤ (F ⊓ v ⊓ e * −F) * (F ⊓ −F * v) * F
by (metis asms(1) inf.absorb1 inf.cobounded1 mult-isotone preorder-ideal-potent)
also have ... ≤ (F ⊓ v ⊓ e * −F) * (F ⊓ −F * v) * (F ⊓ v) * T
using asms(5) comp-inf.mult-right-isotone by auto
also have ... ≤ (−F ⊓ vT) ∗ −F ∗ −F ∗ (−F ⊓ v) ∩ (F ⊓ v)T* ∗ (F ⊓ v)*
proof −
also have ... = (vT ∩ F ⊓ −F) ∗ −F
  by (metis conv-complement dedekind-2 inf-commute)
also have ... = (vT ∩ F ⊓ −F) ∗ −F
  using assms(1) equivalence-comp-left-complement by simp
finally have F ∩ vT * −F ≤ F ∩ (vT ∩ −F) ∗ −F
  using assms(1) by auto
hence 11: F ∩ vT ∗ −F = F ∩ (−F ∩ vT) ∗ −F
  by (metis inf.antisym-cone inf.sup-monoid.add-commute comp-left-subdist-inf inf. boundedE inf.sup-right-isotone)
  by (metis (full-types) assms(1) conv-complement conv-dist-comp conv-dist-inf)
  by (simp add: abel-semigroup.axioms)
also have ... ≤ (−F ∩ vT) ∗ −F ∗ −F ∗ (−F ∩ v)
  by (metis comp-associative comp-isotone inf.cobounded2)
finally show ?thesis
  using comp-inf.mult-left-isotone by blast
qed
also have ... = ((−F ∩ vT) ∗ −F ∗ −F ∗ (−F ∩ v) ∩ (F ∩ v)T* ∗ (F ⊓ v)*)
  ⊓ (−F ∩ vT) ∗ −F ∗ −F ∗ (−F ∩ v) ∩ (F ∩ v)*
  by (metis comp-associative inf-sup-distrib1 star.circ-loop-fixpoint)
also have ... = ((−F ∩ vT) ∗ −F ∗ −F ∗ (−F ∩ v) ∩ (F ∩ vT) ∗ (F ⊓ v)*)
  ⊓ (−F ∩ vT) ∗ −F ∗ −F ∗ (−F ∩ v) ∩ (F ∩ v)*)
  by (simp add: conv-dist-inf) by auto
also have ... = bot ∪ ((−F ∩ vT) ∗ −F ∗ −F ∗ (−F ∩ v) ∩ (F ∩ v)*)
proof −
  have (−F ∩ vT) ∗ −F ∗ −F ∗ (−F ∩ v) ∩ (F ∩ vT) ∗ (F ⊓ v)T* ∗ (F ⊓ v) 
    ≤ bot
    using assms(1, 2) forests-bot-2 by (simp add: comp-associative)
    thus ?thesis
    using le-bot by blast
qed
also have ... = (−F ∩ vT) ∗ −F ∗ −F ∗ (−F ∩ v) ∩ (1 ∪ (F ∩ v)*) ∗ (F ⊓ v)
  by (simp add: star.circ-plus-same star-left-unfold-equal)
also have ... = (−F ∩ vT) ∗ −F ∗ −F ∗ (−F ∩ v) ∩ (1 ∪ (−F ∩ vT) ∗ −F ∗ −F ∗ (−F ∩ v) ∩ (F ∩ v)*)
  by (simp add: comp-inf.semiring.distrib-left)
also have ... ≤ 1 ∪ ((−F ∩ vT) ∗ −F ∗ −F ∗ (−F ∩ v) ∩ (F ⊓ v)*) ∗ (F ⊓ v)
  using sup-left-isotone by auto
also have ... ≤ 1 ∪ bot
using assms(1, 2) forests-bot-3 comp-inf.semiring.add-left-mono by simp

finally show \( \text{thesis} \)

by simp

qed

have 2: \( ?i \ast \text{top} \ast ?i^T \leq 1 \)

proof -

have \( ?i \ast \text{top} \ast ?i^T = (v \sqcap -F \ast e \ast \text{top} \sqcap \text{top} \ast e^T \ast F) \ast \text{top} \ast (v^T \sqcap (-F \ast e \ast \text{top})) \sqcap (\text{top} \ast e^T \ast F)^T \)

by (simp add: conv-dist-inf)

also have \( = (v \sqcap -F \ast e \ast \text{top} \sqcap \text{top} \ast e^T \ast F) \ast \text{top} \ast (v^T \sqcap \text{top} \ast e^T \ast -F^T \sqcap F \ast e^T \ast \text{top}) \sqcap \text{top} \ast e^T \ast -F \)

by (simp add: conv-complement conv-dist-comp mult-assoc)

also have \( = (v \sqcap -F \ast e \ast \text{top} \sqcap \text{top} \ast e^T \ast F) \ast \text{top} \ast (v^T \sqcap \text{top} \ast e^T \ast -F \sqcap F \ast e \ast \text{top}) \)

by (simp add: assms(1))

also have \( = -F \ast e \ast \text{top} \sqcap (v \sqcap \text{top} \ast e^T \ast F) \ast \text{top} \ast (v^T \sqcap \text{top} \ast e^T \ast -F \sqcap F \ast e \ast \text{top}) \sqcap \text{top} \ast e^T \ast -F \)

by (smt inf.left-commute inf.sup-monoid.add-assoc vector-export-comp)

also have \( = -F \ast e \ast \text{top} \sqcap (v \sqcap \text{top} \ast e^T \ast F) \ast \text{top} \ast (v^T \sqcap F \ast e \ast \text{top}) \sqcap \text{top} \ast e^T \ast -F \)

by (smt comp-inf-covector inf.sup-monoid.add-assoc inf.sup-monoid.add-commute mult-assoc)

also have \( = -F \ast e \ast \text{top} \sqcap (v \sqcap \text{top} \ast e^T \ast F) \ast \text{top} \ast (v^T \sqcap F \ast e \ast \text{top}) \sqcap \text{top} \ast e^T \ast -F \)

by (simp add: mult-assoc)

also have \( = -F \ast e \ast \text{top} \sqcap (\text{top} \ast e^T \ast F)^T \sqcap \text{top} \ast (v^T \sqcap F \ast e \ast \text{top}) \sqcap \text{top} \ast e^T \ast -F \)

by (simp add: comp-inf-vector-1 mult.semidom-axioms semigroup.assoc)

also have \( = -F \ast e \ast \text{top} \sqcap (\text{top} \ast e^T \ast F)^T \sqcap \text{top} \ast (\text{top} \ast (F \ast e \ast \text{top}) \ast v^T \sqcap \text{top} \ast e^T \ast -F \)

by (smt comp-inf-vector-covector-comp-inf vector-mult-closed vector-top-closed)

also have \( = -F \ast e \ast \text{top} \sqcap (v \ast e \ast \text{top} \ast (v \ast e \ast \text{top}) \ast (F \ast e \ast \text{top}) \ast v^T \sqcap \text{top} \ast e^T \ast -F \)

by simp

also have \( = -F \ast e \ast \text{top} \sqcap (v \ast F \ast e \ast \text{top} \ast \text{top} \ast \text{top} \ast e^T \ast F \ast e \ast \text{top} \ast \text{top} \ast e^T \ast -F \)

by (metis comp-associative conv-dist-comp)

also have \( = -F \ast e \ast \text{top} \sqcap (v \ast F \ast e \ast \text{top} \ast \text{top} \ast \text{top} \ast e^T \ast F \ast v^T \sqcap \text{top} \ast e^T \ast -F \)

by (auto)

also have \( = -F \ast e \ast \text{top} \sqcap (v \ast F \ast e \ast \text{top} \ast \text{top} \ast e^T \ast F \ast v^T \sqcap \text{top} \ast e^T \ast -F \)

by (smt comp-associative comp-inf-covector inf.sup-monoid.add-assoc inf.top.left-neutral vector-top-closed)

also have \( = (-F \sqcap v \ast F) \ast e \ast \text{top} \sqcap \text{top} \ast e^T \ast F \ast v^T \sqcap \text{top} \ast e^T \ast -F \)

using injective-comp-right-dist-inf assms(3) mult.semidom-axioms semigroup.assoc by fastforce

also have \( = (-F \sqcap v \ast F) \ast e \ast \text{top} \sqcap \text{top} \ast e^T \ast (F \ast v^T \sqcap -F) \)

59
\textbf{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
al also have ... \((F \cap v * F) * e * \top * \top * e^T * (F * v^T \sqcap -F)\)
by (metis inf-top-right vector-export-comp vector-top-closed)
also have ... \((F \cap v * F) * e * \top * e^T * (F * v^T \sqcap -F)\)
by (simp add: comp-associative)
also have ... \(\leq (F \cap v * F) * (F * v^T \sqcap -F)\)
by (smt assms(3) conv-dist-comp mult.semigroup-axioms mult-left-isotope shunt-bijective symmetric-top-closed top-right-mult-increasing semigroup.assoc)
also have ... \((F \cap v * F) * ((v * F)^T \sqcap -F)\)
by (simp add: assms(1) conv-dist-comp)
also have ... \((F \cap v * F) * (-F \sqcup v * F)^T\)
using assms(1) conv-complement conv-dist-inf by (simp add:
inf.sup-monoid.add-commute)
also have ... \(\leq (F \cap v) * (F \cap v) * (F \cap v)^T * (F \cap v)^T\)
proof –
\begin{align*}
\text{let } ?Fv &= F \cap v \\
\text{have } -F \cap v * F &\leq (F \cap v) * (F \cap v)^T * (F \cap v)^T \\
\text{using assms(5) inf.sup-right-isotope mult-right-isotope comp-associative by auto}
\end{align*}
al also have ... \(\leq -F \cap v * (F \cap v)^T\)
proof –
\begin{align*}
\text{have } v * v^T &\leq 1 \\
\text{by (simp add: assms(2))} \\
\text{hence } v * v^T &\leq F \\
\text{using assms(1) dual-order.trans mult-left-isotope by blast} \\
\text{hence } v * v^T &\leq F \\
\text{by (metis assms(1) mult-1-right preorder-idempotent)}
\end{align*}
star.circ-sup-one-right-unfold star.circ-transitive-equal star-one
star-simulation-right-equal mult-assoc
\begin{align*}
\text{hence } v &* (F \cap v)^T * (F \cap v)^T * F^* \leq F \\
\text{by (meson conv-isotope dual-order.trans inf.cobounded2)} \\
\text{inf.sup-monoid.add-commute mult-left-isotope mult-right-isotope)} \\
\text{hence } v &* (F \cap v)^T * (F \cap v)^T * (F \cap v)^* \leq F \\
\text{by (meson conv-isotope dual-order.trans inf.cobounded2)}
\end{align*}
inf.sup-monoid.add-commute mult-right-isotope mult-right-isotope comp-isotope
conv-dist-inf inf.cobounded1 star-isotope)
\begin{align*}
\text{hence } -F \cap v * (F \cap v)^T * (F \cap v)^T * (F \cap v)^* &\leq \bot \\
\text{using eq-iff p-antitone pseudo-complement by auto} \\
\text{hence } (-F \cap v * (F \cap v)^T * (F \cap v)^T * (F \cap v)^* \sqcup v * (v \cap F)^*) &\leq v \\
\text{using bot-least le-bot by fastforce} \\
\text{hence } (-F \cup v * (F \cap v)^* \sqcup (v * (F \cap v)^T * (F \cap v)^T * (F \cap v)^* \sqcup v \\
\text{by (simp add: sup-inf-distrib2)} \\
\text{hence } (-F \cup v * (F \cap v)^* \sqcup ((F \cap v)^T * (F \cap v)^T * \sqcup I) * (v \cap F)^* &\leq v * (v \cap F)^* \\
\text{by (simp add: inf.sup-monoid.add-commute mult.semigroup-axioms
\end{align*}
\textbf{mult-left-dist-sup \ mult-right-dist-sup semigroup.assoc} \\
\textbf{hence} \ ((-F \sqcup v \ast (v \sqcap F)^\ast) \sqcap v \ast (F \sqcap v)^T) \ast (v \sqcap F)^\ast \leq v \ast (v \sqcap F)^\ast \\
\textbf{by (simp add: star-left-unfold-equal sup-commute)} \\
\textbf{hence} \ (-F \sqcap v \ast (F \sqcap v)^T) \ast (v \sqcap F)^\ast \leq v \ast (v \sqcap F)^\ast \\
\textbf{using comp-inf.\ mult-right-sub-dist-sup-left inf.order-lesseq-imp by blast} \\
\textbf{thus} \ ?thesis \\
\textbf{by (simp add: inf.sup-monoid.add-commute)} \\
\textbf{qed} \\
\textbf{also have} \ ... \leq (v \sqcap -F \ast (F \sqcap v)^T) \ast (F \sqcap v)^\ast \\
\textbf{by (metis dedekind-2 cone-star-commute inf.sup-monoid.add-commute)} \\
\textbf{also have} \ ... \leq (v \sqcap -F \ast F^T) \ast (F \sqcap v)^\ast \\
\textbf{using conv-isotone inf.sup-right-isotone mult-left-isotope mult-right-isotope} \\
\textbf{star-isotope by auto} \\
\textbf{also have} \ ... = (v \sqcap -F \ast F) \ast (F \sqcap v)^\ast \\
\textbf{by (metis assms(1) equivalence-comp-right-complement mult-left-one} \\
\textbf{star-one star-simulation-right-equal}) \\
\textbf{also have} \ ... = (-F \sqcap v) \ast (F \sqcap v)^\ast \\
\textbf{using assms(1) equivalence-comp-right-complement} \\
\textbf{inf.sup-monoid.add-commute by auto} \\
\textbf{finally have} \ (-F \sqcap v \ast F) \leq (-F \sqcap v) \ast (F \sqcap v)^\ast \\
\textbf{by simp} \\
\textbf{hence} \ (-F \sqcap v \ast F) \leq (-F \sqcap v) \ast (F \sqcap v)^\ast \ast ((-F \sqcap v \ast (F \sqcap v)^\ast) \sqcap (v \sqcap F)^\ast) \sqcap (F \sqcap v)^T \\
\textbf{by (simp add: comp-isotope conv-isotone}) \\
\textbf{also have} \ ... = (-F \sqcap v) \ast (F \sqcap v)^\ast \ast (F \sqcap v)^T \ast (F \sqcap v)^\ast \\
\textbf{by (simp add: comp-associative conv-dist-comp conv-star-commute}) \\
\textbf{finally show} \ ?thesis \\
\textbf{by simp} \\
\textbf{qed} \\
\textbf{also have} \ ... \leq (F \sqcap v)^\ast \ast ((F \sqcap v)^\ast \sqcup (F \sqcap v)^T) \\
\textbf{proof} – \\
\textbf{have} \ (F \sqcap v)^\ast \ast (F \sqcap v)^T \leq F^\ast \ast F^T \\
\textbf{using fc-isotone by auto} \\
\textbf{also have} \ ... \leq F \ast F \\
\textbf{by (metis assms(1) preorder-idempotent star.circ-sup-one-left-unfold} \\
\textbf{star.circ-transitive-equal star-right-induct-mult}) \\
\textbf{finally have} \ 21: \ (F \sqcap v) \ast (F \sqcap v)^T \leq F \\
\textbf{using assms(1) dual-order.trans by blast} \\
\textbf{have} \ (F \sqcap v)^\ast \ast (F \sqcap v)^T \leq v^\ast \ast v^T \\
\textbf{by (simp add: fc-isotone}) \\
\textbf{hence} \ (F \sqcap v)^\ast \ast (F \sqcap v)^T \leq F \sqcap v^\ast \ast v^T \\
\textbf{using 21 by simp} \\
\textbf{also have} \ ... = F \sqcap (v^\ast \sqcup v^T) \\
\textbf{by (simp add: assms(2) cancel-separate-eq}) \\
\textbf{finally show} \ ?thesis \\
\textbf{by (metis assms(4, 6) comp-associative comp-inf,semiring,distrib-left} \\
\textbf{comp-isotope inf-pp-semi-commute mult-left-isotope regular-closed-inf}) \\
\textbf{qed} \\
\textbf{also have} \ ... \leq (-F \sqcap v) \ast (F \sqcap v^T) \ast (-F \sqcap v)^T \sqcup (-F \sqcap v) \ast (F \sqcap v)^\ast \\

61
\[ (F \cap v)^T \]

by (simp add: mult-left-dist-sup mult-right-dist-sup)

also have \( \ldots \leq (F \cap v) \cap (F \cap v)^T \cap (F \cap v) \cap (F \cap v)^T \)

proof –

\[ (F \cap v) \cap (F \cap v)^T \leq (F \cap v) \cap (F \cap v)^T \]

by (simp add: assms(5) inf.coboundedI1 mult-right-isotone)

also have \( \ldots \leq (F \cap v) \cap (F \cap v)^T \cap (F \cap v)^T \cap (F \cap v)^T \)

by (metis comp-associative comp-inf\.mult-right-dist-sup mult-left-dist-sup inf-sup-monoid.add-commute le-supI1)

also have \( \ldots \leq (F \cap v) \cap (F \cap v)^T \cap (F \cap v) \cap (F \cap v)^T \)

by (simp add: comp-associative comp-inf\.inf subpoisson.add-monoseq eq-iff)

also have \( \ldots \leq bot \cup (F \cap v) \cap (v^* \cap v^T^*) \)

by (metis assms(1, 2) forests-bot-1 comp-associative)

also have \( \ldots \leq bot \cup (F \cap v) \cap (v^* \cap v^T^*) \)

by (simp add: \ldots )

finally have \( 22: (F \cap v) \cap (F \cap v)^T \leq (F \cap v)^T \)

using \( 22 \) conv-isotone by fastforce
thus \( \vdash \)

using \( 22 \) by (metis assms(4, 6) comp-associative)

qed
also have ... = (top ∩ (−F ⊗ e ⊗ top)ᵀ) ⊗ v ⊗ ((top ⊗ eᵀ ⊗ F)ᵀ ∩ top)
   using comp-inf-vector conv-dist-comp by auto
also have ... = (−F ⊗ e ⊗ top)ᵀ ⊗ v ⊗ (top ⊗ eᵀ ⊗ F)ᵀ
   by simp
also have ... = topᵀ ⊗ eᵀ ⊗ −Fᵀ ⊗ v ⊗ Fᵀ ⊗ eᵀᵀ ⊗ topᵀ
   by (simp add: comp-associative conv-complement conv-dist-comp)
finally have 33: top ∗ ?i ∗ top = top ∗ eᵀ ∗ −F ⊗ v ⊗ F ⊗ e ⊗ top
   by (simp add: assms(1))
have top ∗ ?i ∗ top ≠ bot
proof (rule ccontr)
  assume ¬ top ∗ (v ⊓ −F ⊗ e ⊗ top ⊓ top ∗ eᵀ ⊗ F) ∗ top ≠ bot
  hence top ∗ eᵀ ∗ −F ⊗ v ⊗ F ⊗ e ⊗ top = bot
  using 33 by auto
  hence eᵀ ⊓ e = bot
  using 31 tarski comp-associative le-bot by fastforce
  hence top ∗ (−F ⊗ v ⊗ F ⊗ e)ᵀ ⊓ −(eᵀ)
    by (metis comp-associative conv-complement-sub-leq conv-involutive p-bot)
schroeder-5-p
  hence top ∗ eᵀ ∗ Fᵀ ⊗ vᵀ ⊓ −Fᵀ ≤ −(eᵀ)
    by (simp add: comp-associative conv-complement conv-dist-comp)
hence v ⊗ F ⊗ e ⊗ top ⊓ eᵀ ≤ F
    by (metis assms(1, 4) comp-associative conv-dist-comp
symmetric-top-closed)
  hence v ⊗ F ⊗ e ⊗ top ⊓ top ⊗ eᵀ ≤ F
    by (simp add: comp-associative)
hence v ⊗ F ⊗ e ⊗ top ≤ F ⊗ (top ⊗ eᵀ)ᵀ
    using 32 by (metis shunt-bijective
      comp-associative conv-involutive)
hence v ⊗ F ⊗ e ⊗ top ≤ F ⊗ e ⊗ top
    using comp-associative conv-dist-comp by auto
  hence v ∗ ⊗ F ⊗ e ⊗ top ≤ F ⊗ e ⊗ top
    using comp-associative star-left-induct-mult-iff by auto
  hence eᵀ ⊗ F ⊗ e ⊗ top ≤ F ⊗ e ⊗ top
    by (meson assms(9) mult-left-isotone order-trans)
hence eᵀ ⊗ F ⊗ e ⊗ top ⊓ (e ⊗ top)ᵀ ≤ F
    using 32 shunt-bijective assms(3) mult-assoc by auto
  hence 34: eᵀ ⊗ F ⊗ e ⊗ top ⊓ top ⊗ eᵀ ≤ F
    by (metis conv-dist-comp mult.semigroup-axioms symmetric-top-closed
symmetric-top-closed)
  hence eᵀ ≤ F
proof –
  have eᵀ ≤ eᵀ ⊗ e ⊗ eᵀ
    by (metis conv-involutive ex231c)
  also have ... ≤ eᵀ ⊗ F ⊗ e ⊗ eᵀ
    using assms(1) comp-associative mult-left-isotone mult-right-isotone by fastforce
also have ... ≤ eᵀ ⊗ F ⊗ e ⊗ top ⊓ top ⊗ eᵀ
    by (simp add: mult-left-isotone top-right-mult-increasing
vector-mul-closed)
finally show ?thesis

63
using 34 by simp

qed

hence 35: \( e \leq F \)

using assms(1) conv-order by fastforce

have \( \text{top} \ast (F \ast e)^T \leq -e \)

using assms(8) comp-associative schroeder-4-p by auto

hence \( \text{top} \ast e^T \ast F \leq -e \)

by (simp add; assms(1) comp-associative conv-dist-comp)

hence \( (\text{top} \ast e^T)^T \ast e \leq -F \)

using schroeder-3-p by auto

hence \( e \ast \text{top} \ast e \leq -F \)

by (simp add; conv-dist-comp)

hence \( e \leq -F \)

by (simp add; assms(3) arc-top-arc)

hence \( e \leq F \land -F \)

using 35 inf. boundedI by blast

hence \( e = \text{bot} \)

using bot-unique by auto

thus \( \text{False} \)

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 \ast \text{top}) \land bijective (?i^T \ast \text{top})

using 1 2 3 arc-expanded by blast

thus ?thesis

by blast

qed

4.3.3 Comparison of edge weights

In this section we compare the weight of the selected-edge with other edges of interest. Theorems 8, 9, 10 and 11 are supporting lemmas. For example, Theorem 8 is used to show that the selected-edge has its source inside and its target outside the component it is chosen for.

Theorem 8

lemma e-leq-c-c-complement-transpose-general:

assumes \( e = \text{minarc} (c \ast -(c)^T \sqcap g) \)

and regular \( c \)

shows \( e \leq c \ast -(c)^T \)

proof –

have \( e \leq -\quad (c \ast -(c)^T \sqcap g) \)

using assms(1) minarc-below order-trans by blast

also have \( \ldots \leq -\quad (c \ast -(c)^T) \)

using order-lesseq-imp pp-isotone-inf by blast

also have \( \ldots = c \ast -(c)^T \)
using assms(2) regular-mult-closed by auto
finally show ?thesis
  by simp
qed

Theorem 9

lemma x-leq-c-transpose-general:
  assumes forest h and vector c
  and $x^T \cdot \top \leq \text{forest-components}(h) \cdot c \cdot \top$
  and $e \leq c \cdot -c^T$
  and $c = \text{forest-components}(h) \cdot c$
  shows $x \leq c^T$
proof
  let $?H = \text{forest-components } h$
  have $x \leq \top \cdot x$
    using top-left-mult-increasing by blast
  also have $\ldots \leq (?H \cdot e \cdot \top)^T$
    using assms(3) conv-dist-comp conv-order by force
  also have $\ldots = \top \cdot c^T \cdot ?H$
    using assms(1) comp-associative conv-dist-comp
  forest-components-equivalence by auto
  also have $\ldots \leq \top \cdot (c \cdot -c^T)^T \cdot ?H$
    by (simp add: assms(4) conv-isotone mult-left-isotone mult-right-isotone)
  also have $\ldots = \top \cdot (-c \cdot c^T) \cdot ?H$
    by (simp add: conv-complement conv-dist-comp)
  also have $\ldots \leq \top \cdot c^T \cdot ?H$
    by (metis mult-left-isotone top.extremum mult-assoc)
  also have $\ldots = c^T \cdot ?H$
    using assms(1, 2) component-is-vector vector-conv-covector by auto
  also have $\ldots = c^T$
    by (metis assms(1, 5) fch-equivalence conv-dist-comp)
  finally show ?thesis
    by simp
qed

Theorem 10

lemma x-leq-c-complement-general:
  assumes vector c
  and $c \cdot c^T \leq \text{forest-components } h$
  and $x \leq c^T$
  and $x \leq -\text{forest-components } h$
  shows $x \leq -c$
proof
  let $?H = \text{forest-components } h$
  have $x \leq -？H \cap c^T$
    using assms(3, 4) by auto
  also have $\ldots \leq -c$
    proof
have $c \cap c^T \leq ?H$
  using assms\{1, 2\} vector-covector by auto
hence $-?H \cap c \cap c^T \leq \bot$
  using inf_sup-monoid.add-assoc p-antitone pseudo-complement by fastforce
thus $\neg \thesis$
  using le-bot p-shunting-swap pseudo-complement by blast
qed
finally show $\neg \thesis$
  by simp
qed

Theorem 11

Lemma: sum-e_below_sum-x_when_outgoing_same_component_general:

assumes $e = \minarc (c * -(c)^T \cap g)$
  and regular $c$
  and forest $h$
  and vector $c$
  and $x^T * \top \leq (\text{forest-components } h) * e * \top$
  and $c = (\text{forest-components } h) * c$
  and $c * c^T \leq \text{forest-components } h$
  and $x \leq -\text{forest-components } h \cap -- g$
  and symmetric $g$
  and arc $x$
  and $c \neq \bot$
shows $\text{sum } (e \cap g) \leq \text{sum } (x \cap g)$

proof –
let $?H = \text{forest-components } h$

have 1: $e \leq c * -(c)^T$
  using assms\{1, 2\} e-leq-c-c-complement-transpose-general by auto

have 2: $x \leq c^T$
  using 1 assms\{3, 4, 5, 6\} x-leq-c-transpose-general by auto
hence $x \leq -c$
  using assms\{4, 7, 8\} x-leq-c-complement-general inf boundedE by blast
hence $x \leq -c \cap c^T$
  using 2 by simp
hence $x \leq -c * c^T$
  using assms\{4\} by (simp add: vector-complement-closed vector-covector)
hence $x^T \leq c^{TT} * -(c)^T$
  by (metis conv-complement conv-dist-comp conv-isotone)
hence 3: $x^T \leq c * -(c)^T$
  by simp
hence $x \leq -- g$
  using assms\{8\} by auto
hence $x^T \leq -- g$
  using assms\{9\} conv-complement conv-isotone by fastforce
hence $x^T \cap c * -(c)^T \cap -- g \neq \bot$
  using 3 by (metis assms\{10, 11\} comp-inf.semiring.mult-not-zero
  conv-dist-comp
  conv-involutive inf.orderE mult-right-zero top.extremum)
hence \( x^T \cap c * = c^T \cap g \neq bot \)
  using inf.sup-monoid.add-commute pp-inf-bot-iff by auto

hence \( \sum (\text{minarc} (c * - c^T \cap g) \cap (c * - c^T \cap g)) \leq \sum (x^T \cap c * - c^T \cap g) \)
  using assms(10) minarc-min inf.sup-monoid.add-assoc by auto

hence \( \sum (e \cap c * - c^T \cap g) \leq \sum (x^T \cap c * - c^T \cap g) \)
  using 1 3 by (metis inf.orderE)

hence \( \sum (e \cap g) \leq \sum (x \cap g) \)
  using assms(9) sum-symmetric by auto

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 \( x \leq - \text{forest-components} h \cap -g \)
  and \( x^T * \text{top} \leq \text{forest-components} h * \text{selected-edge} h j g * \text{top} \)
  and \( j \neq bot \)
  and arc x

shows \( \sum (\text{selected-edge} h j g \cap g) \leq \sum (x \cap g) \)

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-x-when-outgoing-same-component-general)
  next
    show \(?e\) = minarc (?c * - ?e^T \cap g)
      by simp
  next
    show regular ?c
      using component-is-regular by auto
  next
    show forest h
      by (simp add: assms(3))
  next
    show vector ?c
      by (simp add: assms(2, 6) component-is-vector)
  next
    show \( x^T * \text{top} \leq ?H * ?e * \text{top} \)
      by (simp add: assms(5))
  next
    show ?c = ?H * ?c
      using component-single by auto
  next

show \(?c \ast ?c^T \leq ?H\)
by (simp add: component-is-connected)

next
show \(x \leq \sim ?H \cap \sim g\)
using assms(4) by auto

next
show symmetric \(g\)
by (simp add: assms(1))

next
show \(\text{arc}\ x\)
by (simp add: assms(7))

next
show \(?c \neq \text{bot}\)
using assms(2,5,6,7) inf-bot-left le-bot minarc-bot mult-left-zero
mult-right-zero by fastforce

qed

qed

If there is a path in the big-forest 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 \(\text{sum-e-below-sum-x-when-outgoing-same-component}\), 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 \(\text{big-forest-path-split-disj}\). We show that the weight of \(e\) is no greater than the weight of \(x\) by making use of lemma \(\text{sum-e-below-sum-x-when-outgoing-same-component}\). 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-bigforest:
assumes symmetric \(g\)
and \(f \leq \sim g\)
and vector \(j\)
and forest \(h\)
and \(\text{big-forest} (\text{forest-components}\ h)\ d\)
and \(f \uplus f^T = h \uplus h^T \uplus d \uplus d^T\)
and \((\forall a\ b. \text{bf-between-arcs}\ a\ b\ (\text{forest-components}\ h)\ d \land a \leq \sim (\text{forest-components}\ h) \cap \sim g \land b \leq d \longrightarrow \text{sum}(b \cap g) \leq \text{sum}(a \cap g))\)
and regular \(d\)
and \(j \neq \text{bot}\)
and \(b = \text{selected-edge}\ h\ j\ g\)
and arc \(a\)
and \(\text{bf-between-arcs}\ a\ b\ (\text{forest-components}\ h)\ (d \uplus \text{selected-edge}\ h\ j\ g)\)
and \(a \leq \sim (\text{forest-components}\ h) \cap \sim g\)
and regular h
shows \( \text{sum} (b \cap g) \leq \text{sum} (a \cap g) \)
proof -
let \(?p = \text{path } f \  h \  j \  g\)
let \(?e = \text{selected-edge } h \  j \  g\)
let \(?F = \text{forest-components } f\)
let \(?H = \text{forest-components } h\)
have \( \text{sum} (b \cap g) \leq \text{sum} (a \cap g) \)
proof (cases \( a^T \ast \top \leq ?H \ast ?e \ast \top \))
  case True
  show \( a^T \ast \top \leq ?H \ast ?e \ast \top \implies \text{sum} (b \cap g) \leq \text{sum} (a \cap g) \)
  proof
    have \( \text{sum} (\ ?e \cap g) \leq \text{sum} (\ a \cap g) \)
    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(2) by blast
    next
    show forest h
      by (simp add: assms(4))
    next
    show \( a \leq - ?H \cap - - g \)
      using assms(13) by auto
    next
    show \( a^T \ast \top \leq ?H \ast ?e \ast \top \)
      using True by auto
    next
    show \( j \neq \bot \)
      by (simp add: assms(9))
    next
    show arc a
      by (simp add: assms(11))
  qed
  thus \( ?\text{thesis} \)
    using assms(10) by auto
  qed
next
case False
show \( \neg a^T \ast \top \leq ?H \ast ?e \ast \top \implies \text{sum} (b \cap g) \leq \text{sum} (a \cap g) \)
proof -
let \(?d' = d \cup ?e\)
let \(?x = d \cap \top \ast ?e^T \ast ?H \cap (?H \ast d^T)^* \ast ?H \ast a^T \ast \top\)
have 61: arc \(?x\)
proof (rule shows-arc-x)
  show big-forest \(?H \ d\)
    by (simp add: assms(5))
next
show \( \text{bf-between-arcs} \, a \, ?e \, ?H \, d \)

proof
  have 611: \( \text{bf-between-arcs} \, a \, b \, ?H \, (d \sqcup b) \)
    using \( \text{assms}(10, \, 12) \) by auto
  have 616: \( \text{regular} \, h \)
    using \( \text{assms}(14) \) by auto
  have \( \text{regular} \, a \)
    using \( \text{bf-between-arcs-def} \, \text{arc-regular} \) by fastforce
  thus \( \text{thesis} \)
    using 616 by \( \text{smt big-forest-path-split-disj} \, \text{assms}(4, \, 8, \, 10, \, 12) \)
    \( \text{bf-between-arcs-def} \, \text{fch-equivalence} \, \text{minarc-regular} \, \text{regular-closed-star} \)
    \( \text{regular-conv-closed} \, \text{regular-mult-closed} \)
  qed
next
  show \( \langle ?H \, \ast \, d \rangle^* \leq - \, ?H \)
    using \( \text{assms}(5) \) \( \text{big-forest-def} \) by blast
next
  show \( \neg \, a^T \, \ast \, \text{top} \leq \, ?H \, \ast \, ?e \, \ast \, \text{top} \)
    by (simp add: False)
next
  show \( \text{regular} \, a \)
    using \( \text{assms}(12) \) \( \text{bf-between-arcs-def} \, \text{arc-regular} \) by auto
next
  show \( \text{regular} \, ?e \)
    using \( \text{minarc-regular} \) by auto
next
  show \( \text{regular} \, ?H \)
    using \( \text{assms}(14) \) \( \text{pp-dist-star} \, \text{regular-conv-closed} \, \text{regular-mult-closed} \) by auto
next
  show \( \text{regular} \, d \)
    using \( \text{assms}(8) \) by auto
  qed
have 62: \( \text{bijective} \, (a^T \, \ast \, \text{top}) \)
  by (simp add: \( \text{assms}(11) \))
have 63: \( \text{bijective} \, (?x \, \ast \, \text{top}) \)
  using 61 by simp
have 64: \( ?x \leq (\langle ?H \, \ast \, d^T \rangle^* \, \ast \, ?H \, \ast \, a^T \, \ast \, \text{top}) \)
  by simp
  hence \( ?x \, \ast \, \text{top} \leq (\langle ?H \, \ast \, d^T \rangle^* \, \ast \, ?H \, \ast \, a^T \, \ast \, \text{top}) \)
    using \( \text{mult-left-isotone} \, \text{inf-vector-comp} \) by auto
  hence \( a^T \, \ast \, \text{top} \leq (\langle ?H \, \ast \, d^T \rangle^* \, \ast \, ?H)^T \, \ast \, ?x \, \ast \, \text{top} \)
    using 62 63 64 by \( \text{smt bijective-reverse} \, \text{mult-assoc} \)
also have \( \ldots \, = \langle ?H \, \ast \, (d \, \ast \, ?H) \rangle^* \, \ast \, ?x \, \ast \, \text{top} \)
  using \( \text{conv-dist-comp} \, \text{conv-star-commute} \) by auto
also have \( \ldots \, = (\langle ?H \, \ast \, d \rangle^* \, \ast \, ?H \, \ast \, ?x \, \ast \, \text{top}) \)
  by (simp add: star-slide)
finally have \( a^T \, \ast \, \text{top} \leq (\langle ?H \, \ast \, d \rangle^* \, \ast \, ?H \, \ast \, ?x \, \ast \, \text{top}) \)
  by simp

70
hence 65: \( \text{bf-between-arcs } a \ ?x \ ?H \ d \)
using 61 assms(12) \( \text{bf-between-arcs-def} \) by blast
have 66: \( ?x \leq d \)
  by (simp add: inf.sup-monoid.add-assoc)
hence \( \text{x-below-a} \): \( \text{sum} \ (\ ?x \cap g) \leq \text{sum} \ (a \cap g) \)
have \( \text{sum} \ (\ ?e \cap g) \leq \text{sum} \ (\ ?x \cap g) \)
proof (rule \( \text{sum-e-below-sum-x-when-outgoingSame-component} \))
  show symmetric \( g \)
    using assms(1) by auto
next
  show \( \text{vector } j \)
    using assms(3) by blast
next
  show \( \text{forest } h \)
    by (simp add: assms(4))
next
  show \( ?x \leq - ?H \cap -- \ g \)
proof

have 67: \( ?x \leq - ?H \)
  using 66 assms(5) \( \text{big-forest-def} \ \text{order-lesseq-imp} \) by blast
have \( ?x \leq d \)
  by (simp add: conv-isotone inf.sup-monoid.add-assoc)
also have \( \ldots \leq f \sqcup f^T \)
proof

  have \( h \sqcup h^T \sqcup d \sqcup d^T = f \sqcup f^T \)
    by (simp add: assms(6))
  thus \(?\text{thesis}\)
    by (metis (no-types) le-supE sup.absorb_iff2 sup.idem)

  have \( \ldots \leq -- \ g \)
    using assms(1, 2) \( \text{conv-complement} \ \text{conv-isotone} \) by fastforce
finally have \( ?x \leq -- \ g \)
  by simp
  thus \(?\text{thesis}\)
    by (simp add: 67)
qed

next
  show \( ?x^T \ast \text{top} \leq ?H \ast ?e \ast \text{top} \)
proof

  have \( ?x \leq \text{top} \ast ?e^T \ast ?H \)
    using inf.cobounded1I by auto
  hence \( ?x^T \leq ?H \ast ?e \ast \text{top} \)
    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 \( ?x^T \ast \text{top} \leq ?H \ast ?e \ast \text{top} \ast \text{top} \)
    by (simp add: mult-left-isotone)
  thus \(?\text{thesis}\)
    by (simp add: mult-assoc)
qed
next
  show \( j \neq \text{bot} \)
    by (simp add: assms(9))
next
  show \( \text{arc (} ?x \text{)} \)
    using 61 by blast
qed
hence \( \text{sum (} ?e \cap g \text{)} \leq \text{sum (} a \cap g \text{)} \)
  using \( x\)-below-a order.trans by blast
thus \( \text{thesis} \)
  by (simp add: assms(10))
qed
qed
thus \( \text{thesis} \)
  by simp
qed

4.3.4 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 \( \text{exists-a-w} \) to maintain that \( f \) can be extended to a minimum spanning forest.

**lemma** boruvka-exchange-spanning-inv:

assumes \( \text{forest } v \)
and \( v^* \* e^T = e^T \)
and \( i \leq v \cap \text{top} \* e^T \* w^T \)
and \( \text{arc } i \)
and \( \text{arc } e \)
and \( v \leq -\neg g \)
and \( w \leq -\neg g \)
and \( e \leq -\neg g \)
and \( \text{components } g \leq \text{forest-components } v \)
shows \( i \leq (v \cap -i)^T \* e^T \* \text{top} \)

**proof** –

have 1: \( (v \cap -i \cap -i^T) \* (v^T \cap -i \cap -i^T) \leq 1 \)
  using assms(1) comp-isotone order.trans inf.cobounded1 by blast

have 2: bijective \( (i \* \text{top}) \land \text{bijective (} e^T \* \text{top} \)
  using assms(4, 5) mult-assoc by auto

have i \( \leq v \* (\text{top} \* e^T \* w^T)^T \)
  using assms(3) covector-mult-closed covector-restrict-comp-conv

also have \( \text{order-lesseq-imp vector-top-closed by blast} \)

also have \( \leq v \* w^T \* e^T \* \text{top} \)
  by (simp add: comp-associative conv-dist-comp)

also have \( \leq v \* w^* \* e \* \text{top} \)
  by (simp add: conv-star-commute)

also have \( = v \* w^* \* e \* e^T \* e \* \text{top} \)
  using assms(5) arc-eq-1 by (simp add: comp-associative)

also have \( \leq v \* w^* \* e \* e^T \* \text{top} \)

72
by (simp add: comp-associative mult-right-isotone)
also have ... \leq (-(-g) * (-(-g)) * e^T) * top
  using assms(6, 7, 8) by (simp add: comp-isotope star-isotone)
also have ... \leq (-(-g)) * e^T * top
  by (metis comp-isotope mult-left-isotope star.circ-increasing
    star.circ-transitive-equal)
also have ... \leq v^{T\times} * v^* * e^T * top
  by (simp add: assms(9) mult-left-isotope)
also have ... \leq v^{T\times} * e^T * top
  by (simp add: assms(2) comp-associative)
finally have i \leq v^{T\times} * e^T * top
  by simp
hence i * top \leq v^{T\times} * e^T * top
  by (metis comp-associative mult-left-isotope vector-top-closed)
hence e^T * top \leq v^{T\times T} * i * top
  using 2 by (metis bijective-reverse mult-assoc)
also have ... = v^* * i * top
  by (simp add: conv-star-commute)
also have ... \leq (v \sqcap -i \sqcap -i^T) * i * top
proof –
  have 3: i * top \leq (v \sqcap -i \sqcap -i^T)^i * i * top
    using star.circ-loop-fixpoint sup-right-divisibility mult-assoc by auto
  have (v \sqcap i) * (v \sqcap -i \sqcap -i^T)^i * i * top \leq i * top * i * top
    by (metis comp-isotope inf.colbouded1 inf.sup-monoid.add-commute
      mult-left-isotope top.extremum)
  also have ... \leq i * top
    by simp
  finally have 4: (v \sqcap i) * (v \sqcap -i \sqcap -i^T)^i * i * top \leq (v \sqcap -i \sqcap -i^T)^i * i
    * top
    using 3 dual-order.trans by blast
  have 5: (v \sqcap -i \sqcap -i^T)^i * (v \sqcap -i \sqcap -i^T)^i * i * top \leq (v \sqcap -i \sqcap -i^T)^i * i
    * top
    by (metis mult-left-isotope 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-isotope 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-anique pseudo-complement)
  hence 7: v \sqcap i^T \leq bot
    by (metis assms(3) comp-inf.mult-right-isotope conv-dist-inf inf.boundedE
      inf.le-iff-sup le-bot)
  hence (v \sqcap i^T) * (v \sqcap -i \sqcap -i^T)^i * i * top \leq bot
    using le-bot semiring.mult-zero-left by fastforce

73
hence 6: \((v \cap i^T) \ast (v \cap -i \cap -i^T) \ast i \ast \top \leq (v \cap -i \cap -i^T) \ast i \ast \top\) using bot-least le-bot by blast
have 8: \(v = (v \cap i) \cup (v \cap i^T) \cup (v \cap -i \cap -i^T)\)
proof -
have 81: regular i
  by (simp add: assms(4) arc-regular)
have \((v \cap i^T) \cap (v \cap -i \cap -i^T) = (v \cap -i)\)
  using 7 by (metis comp-inf.coreflexive-comp-inf-inf-complement inf-import-p
inf-p le-bot maddux-3-11-pp top.extremum)
hence \((v \cap i) \cup (v \cap i^T) \cup (v \cap -i \cap -i^T) = (v \cap i) \cup (v \cap -i)\)
  by (simp add: sup.semi-group-axioms semigroup.assoc)
also have ... = \(v\)
  using 81 by (metis maddux-3-11-pp)
finally show \(?thesis\)
  by simp
qed

have \((v \cap i) \ast (v \cap -i \cap -i^T) \ast i \ast \top \cup (v \cap i^T) \ast (v \cap -i \cap -i^T) \ast i \ast \top \leq (v \cap -i \cap -i^T) \ast i \ast \top\)
  using 4 5 6 by simp
have \(((v \cap i) \cup (v \cap i^T) \cup (v \cap -i \cap -i^T)) \ast i \ast \top \leq (v \cap -i \cap -i^T) \ast i \ast \top\)
  by (simp add: mult-right-dist-sup)

hence \(v \ast (v \cap -i \cap -i^T) \ast i \ast \top \leq (v \cap -i \cap -i^T) \ast i \ast \top\)
  using 8 by auto
hence \(i \ast \top \cup v \ast (v \cap -i \cap -i^T) \ast i \ast \top \leq (v \cap -i \cap -i^T) \ast i \ast \top\)
  using 3 by auto
hence 9: \(v^* \ast (v \cap -i \cap -i^T) \ast i \ast \top \leq (v \cap -i \cap -i^T) \ast i \ast \top\)
  by (simp add: star-left-induct-mult mult-assoc)
have \(v^* \ast i \ast \top \leq v^* \ast (v \cap -i \cap -i^T) \ast i \ast \top\)
  using 3 mult-right-isotone mult-assoc by auto
thus \(?thesis\)

using 9 order.trans by blast
qed

finally have \(e^T \ast \top \leq (v \cap -i \cap -i^T) \ast i \ast \top\)
  by simp
hence \(i \ast \top \leq (v \cap -i \cap -i^T) \ast e^T \ast \top\)
  using 2 by (metis bijective-reverse mult-assoc)
also have ... = \((v^T \cap -i \cap -i^T) \ast e^T \ast \top\)
  using comp-inf.inf-vector-comp conv-complement-comp conv-dist-inf
conv-star-commute inf_sup-monoid.add-commute by auto
also have ... \(\leq ((v \cap -i \cap -i^T) \cup (v^T \cap -i \cap -i^T)) \ast e^T \ast \top\)
  by (simp add: mult-left-isotone star-isotone)
finally have \(i \leq ((v^T \cap -i \cap -i^T) \cup (v \cap -i \cap -i^T)) \ast e^T \ast \top\)
  using dual-order.trans top-right-mult-increasing sup-commute by auto
also have ... = \((v^T \cap -i \cap -i^T) \ast (v \cap -i \cap -i^T) \ast e^T \ast \top\)
  using 1 cancel-separate-I by (simp add: sup-commute)
also have ... \(\leq (v^T \cap -i \cap -i^T) \ast v^* \ast e^T \ast \top\)
  by (simp add: inf-assoc mult-left-isotone mult-right-isotone star-isotone)
also have ... = \((v^T \cap -i \cap -i^T) \ast e^T \ast \top\)

74
using `assms(2)` `mult-assoc` by simp
also have \( \ldots \leq (v^T \cap -i^T)^* \star e^T \star \top \)
by (metis `mult-left-isotone` `star-isotone` `inf.cobounded2` `inf.left-commute`
`inf.sup-monoid.add-commute`)
also have \( \ldots = (v \cap -i)^T \star e^T \star \top \)
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. \text{minimum-spanning-forest } w g \land f \leq w \cup w^T)`
and `vector j`
and `regular j`
and `forest h`
and `forest-components h \leq forest-components f`
and `big-forest (forest-components h) d`
and `d \star \top \leq - j`
and `forest-components h \star j = j`
and `forest-components f = (forest-components h \star (d \cup d^T))^*` `forest-components h`
and `f \sqcup f^T = h \sqcup h^T \sqcup d \sqcup d^T`
and `(\forall a b. \text{bf-between-arcs } a b (forest-components h) d \land a \leq b)`
and regular `d`
and `selected-edge h j g \leq - forest-components f`
and `selected-edge h j g \neq \text{bot}`
and `j \neq \text{bot}`
and regular `h`
and `h \leq -g`
shows `(\exists w. \text{minimum-spanning-forest } w g \land f \cap - (selected-edge h j g)^T \cap - (\text{path } f h j g) \cup (f \cap - (selected-edge h j g)^T \cap (\text{path } f h j g)^T) \sqcup (selected-edge h j g) \leq w \cup w^T)`

proof –
let `?p = \text{path } f h j g`
let `?e = \text{selected-edge } h j g`
let `?f = (f \cap -?e^T \cap -?p) \cup (f \cap -?e^T \cap ?p)^T \sqcup ?e`
let `?F = \text{forest-components } f`
let `?H = \text{forest-components } h`
let `?ee = \text{choose-component } (\text{forest-components } h) j \star \text{choose-component } (\text{forest-components } h) f^T \cap g`
from `assms(4)` obtain `w` where `2: \text{minimum-spanning-forest } w g \land f \leq w \cup w^T`
using `assms(5)` by blast
hence `3: \text{regular } w \land \text{regular } f \land \text{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 ?eT * top * ?eT = ?eT
  by (metis arc-conv-closed arc-top-arc coreflexive-bot-closed coreflexive-symmetric minarc-arc minarc-bot-iff semiring.mult-not-zero)

hence ?eT * top * ?eT ≤ − ?F
  using 5 assms(17) 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 ⊓ top * ?e * wT ⋆

let ?v = (w ⊓ (top * ?e * wT ⋆)) ⊔ ?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 * wT ⋆)
      by (simp add: covector-mult-closed)
    next
    show top * ?e * wT ⋆ * wT ≤ top * ?e * wT ⋆
      by (simp add: mult-right-isotone star.right-plus-below-circ mult-assoc)
    next
    show coreflexive ((top * ?e * wT ⋆) T * (top * ?e * wT ⋆) T) ∨ wT * 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 ≤ forest-components ?v
  proof (rule kruskal-exchange-spanning-inv-1)
    show injective (w ⊓ (top * ?e * wT ⋆) T ∨ (w ⊓ top * ?e * wT ⋆) T)
      using 8 by simp
    next
    show regular (w ⊓ top * ?e * wT ⋆)
      using 7 by simp
    next
    show components g ≤ 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 conj)
    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 \((top * \?e * w^T)\) 
  by (simp add: covector-mult-closed)
qed
next 
show \(?v \leq - - g\) 
proof \(\text{rule sup-least}\) 
  show \(w \cap -(top * \?e * w^T) \leq - - g\) 
    using 7 inf.coboundedI1 minimum-spanning-forest-def spanning-forest-def 2
  by blast
next 
show \((w \cap top * \?e * w^T)^T \leq - - g\) 
  using 2 by (metis assms(1) 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 \((w \cap -(top * \?e * w^T)) \cap g\) + sum \((?q \cap g)\)\) 
    by (simp add: assms(1) sum-symmetric)
  also have \(... = sum \(((w \cap -(top * \?e * w^T)) \cup \?q) \cap g)\) 
    using inf-commute inf-left-commute sum-disjoint by simp
  also have \(... = sum \((w \cap g)\)\) 
    using 3 7 8 maddux-3-11-pp by auto
finally show \(?thesis\) 
  by simp
qed

have 12: \(?v \sqcup \?v^T = w \sqcup w^T)\)
proof 
  have \(?v \sqcup \?v^T = (w \cap -?q) \sqcup \?q^T \sqcup (w^T \cap -?q^T) \cup \?q)\) 
    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 auto
finally show \(?thesis\) 
  by simp

77
have 13: ?v * ?e^T = bot
proof (rule kruskal-reroot-edge)
  show injective (?e^T * top)
    using assms(18) minarc-arc minarc-bot-iff by blast
next
  show pd-kleene-allegory-classacyclic 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-cone-closed regular-mult-closed by simp

have 16: ?F ≤ − ?i
proof
  have 161: bijective (?e * top)
    using assms(18) 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 * ?e^T * ?F^T * 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
  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

d qed

have 17: \( ?i \leq \top \land ?e^T \ast (?F \land ?v \land \neg ?i)^T \ast \)

proof -

have \( ?i \leq \top \land \neg ?F \ast ?e \ast \top \land \top \ast ?e^T \ast (?F \land ?v)^T \ast \neg (?F \land ?v)^T \ast \neg \)

using 2 11 2 by (smt inf.sup-right-isotone kruskal-forest-components-inf mult-right-isotone mult-assoc)

also have \( \ldots = ?v \land \neg ?F \ast ?e \ast \top \land \top \ast ?e^T \ast (?F \land ?v)^T \ast \neg (1 \sqcup (?F \land ?v)) \)

using star-left-unfold-equal star.circ-right-unfold-1 by auto

also have \( \ldots = (?e \land \neg ?F \ast ?e \ast \top \land (\top \ast ?e^T \ast (?F \land ?v)^T \ast \square \top \ast ?e^T \ast (?F \land ?v)^T \ast \neg (?F \land ?v)) \)

by (simp add: mult-left-dist-sup mult-assoc)

also have \( \ldots = (?e \land \neg ?F \ast ?e \ast \top \land \top \ast ?e^T \ast (?F \land ?v)^T \ast \square \top \ast ?e^T \ast (?F \land ?v)^T \ast \neg (?F \land ?v)) \)

using comp-inf.semiring.distrib-left by blast

also have \( \ldots \leq \top \ast ?e^T \ast (?F \land ?v)^T \ast \square \top \ast ?e^T \ast (?F \land ?v)^T \ast \neg (?F \land ?v) \)

using comp-inf.semiring.add-right-mono inf-le2 by blast

also have \( \ldots \leq \top \ast ?e^T \ast (?F \land ?v)^T \ast \square \top \ast ?e^T \ast (?F \land ?v)^T \ast \neg (?F \land ?v) \)

by (simp add: conv-dist-inf)

also have \( \ldots \leq \top \ast ?e^T \ast (?F \land ?v)^T \ast \square \top \ast ?e^T \ast (?F \land ?v)^T \ast \neg (?F \land ?v) \)

proof -

have \( \top \ast ?e^T \ast (?F \land ?v)^T \ast \square \top \ast ?e^T \ast (?F \land ?v)^T \ast \neg (?F \land ?v) \leq \top \ast ?e^T \ast (?F \land ?v)^T \ast \neg (?F \land ?v) \)

using star-isotone by (simp add: comp-isotone)

hence \( \top \land \neg ?F \ast ?e \ast \top \land \top \ast ?e^T \ast (?F \land ?v)^T \ast \neg (?F \land ?v) \)

using inf.sup-right-isotone by blast

thus \(?\)thesis

using sup-right-isotone by blast

d qed

also have \( \ldots = \top \ast ?e^T \ast (?F \land ?v)^T \ast \square \top \ast ?e^T \ast (?F \land ?v)^T \ast \neg (?F \land ?v) \)

using 5 by auto

also have \( \ldots = \top \ast ?e^T \ast (?F \land ?v)^T \ast \square \top \ast ?e^T \ast (?F \land ?v)^T \ast \neg (?F \land ?v) \)

by (simp add: assms(2) forest-components-star)

also have \( \ldots = \top \ast ?e^T \ast (?F \land ?v)^T \ast \square \top \ast ?e^T \ast (?F \land ?v)^T \ast \neg (?F \land ?v) \)

using 5 mult.semigroup-axioms preorder-idempotent semigroup.assoc by fastforce

also have \( \ldots = \top \ast ?e^T \ast (?F \land ?v)^T \ast \square \top \ast ?e^T \ast (?F \land ?v)^T \ast \neg (?F \land ?v) \)

proof -

have \( ?e \ast \top \ast ?e^T \leq 1 \)

using assms(18) arc-expanded minarc-arc minarc-bot-iff by auto

hence \(?F \ast ?e \ast \top \ast ?e^T \leq ?F \ast 1 \)
by (metis comp-associative comp-isotone mult-semi-associative star.circ-transitive-equal)

hence \(?e \leq 1\) by (simp add: comp-associative conv-dist-inf mult-left-isotone mult-right-isotone)

thus \(?thesis\)

qed

also have \(\cdots\)

by simp

hence \(\cdots\)

by (simp add: comp-associative conv-dist-comp)

also have \(\cdots\)

using 5 by auto

also have \(\cdots\)

using comp-associative by auto

also have \(\cdots\)


finally have \(\cdots\)

by simp

hence \(\cdots\)

by simp

hence \(\cdots\)

by (simp add: le-bot sup-monoid.add-0-right by blast)

also have \(\cdots\)

using 16 by (smt comp-inf.coreflexive-comp-inf-complement inf-top-right p-bot pseudo-complement top.extremum)

finally show \(\cdots\)

by blast

have 18: \(\?i \leq \cdots\)

proof -

have \(\cdots\)

using 17 by simp

also have \(\cdots\)

using mult-right-isotone conv-isotone star-isotone inf.cobounded2
inf.sup-monoid.add-assoc by (simp add: inf.sup-monoid.add-assoc eq-iff inf.sup-monoid.add-commute)
  also have ... ≤ top * ?eT * ((?e ∩ − ?i) ∪ (?e)T)
      using mult-right-isotone conv-isotone star-isotone sup-ge I by simp
  finally show ?thesis by blast
qed

have 19: ?i ≤ top * ?eT * ?vT
  proof
    have ?i ≤ top * ?eT * (?F ∩ (?v ∩ − ?i)T)
      using 17 by simp
    also have ... ≤ top * ?eT * (?v ∩ − ?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 eq-iff inf.sup-monoid.add-commute)
    also have ... ≤ top * ?eT * (?v)T
      using mult-right-isotone conv-isotone star-isotone by auto
    finally show ?thesis by blast
  qed

have 20: f ∪ fT ≤ (?v ∩ − ?vT) ∪ (?vT ∩ − ?vT)
  proof (rule kruskal-edge-between-components-2)
    show ?F ≤ − ?i
      using 16 by simp
  next
    show injective f
      by (simp add: assms(2))
  next
    show f ∪ fT ≤ w ∩ − (top * ?e * wT ∪ (w ∩ top * ?e * wT)T) ∪ (w ∩ − (top * ?e * wT) ∪ (w ∩ top * ?e * wT)T)
      using 2 12 by (metis conv-dist-sup conv-involutive conv-isotone le-sup I sup-commute)
  qed

have minimum-spanning-forest ?w g ∧ ?f ≤ ?w ∩ ?wT
  proof (intro conjI)
    have 211: ?eT ≤ ?wT
      proof (rule kruskal-edge-arc-1 [where g=g and h=?ec])
        show ?e ≤ − − ?ec
          using minarc-below by blast
      next
        show ?ec ≤ g
          using assms(4) inf.cobounded2 by (simp add: borwka-inner-invariant-def borwka-outer-invariant-def conv-dist-inf)
      next
        show symmetric g
          by (meson assms(1) borwka-inner-invariant-def borwka-outer-invariant-def)
      next
        show components g ≤ forest-components (w ∩ − (top * ?e * wT) ∪ (w ∩
\[ \text{top} \ast ?e \ast \text{w}^{\ast T} \]

using \text{9 by simp}

next

\[
\text{show} \quad (\text{w} \cap - (\text{top} \ast ?e \ast \text{w}^{\ast T}) \sqcup (\text{w} \cap \text{top} \ast ?e \ast \text{w}^{\ast T})^T) \ast ?e^T = \text{bot}
\]

using \text{13 by blast}

qed

have \text{212: arc ?i}

proof (rule borueka-edge-arc)

\[
\text{show} \quad \text{equivalence} ?F
\]

by (simp add: 5)

next

\[
\text{show} \quad \text{forest} ?v
\]

using \text{10 spanning-forest-def by blast}

next

\[
\text{show} \quad \text{arc} ?e
\]

using \text{assms(18) minarc-arc minarc-bot-iff by blast}

next

\[
\text{show} \quad \text{regular} ?F
\]

using \text{3 regular-closed-star regular-conv-closed regular-mult-closed by auto}

next

\[
\text{show} \quad ?F \leq \text{forest-components} (\text{?F} \cap ?v)
\]

by (simp add: 12 2 8 kruskal-forest-components-inf)

next

\[
\text{show} \quad \text{regular} ?v
\]

using \text{10 spanning-forest-def by blast}

next

\[
\text{show} \quad ?v \ast ?e^T = \text{bot}
\]

using \text{13 by auto}

next

\[
\text{show} \quad ?e \ast ?F \ast ?e = \text{bot}
\]

by (simp add: 6)

next

\[
\text{show} \quad ?e^T \leq ?v^T
\]

using \text{211 by auto}

next

\[
\text{show} \quad ?e \neq \text{bot}
\]

by (simp add: \text{assms(18)})

qed

show \text{minimum-spanning-forest ?w g}

proof (unfold \text{minimum-spanning-forest-def, intro conjI})

\[
\text{have} \quad (?v \cap - ?i) \ast ?e^T \leq ?v \ast ?e^T
\]

using \text{inf-le mult-left-isotone by simp}

hence \text{221: ?e \ast (?v \cap - ?i)^T = bot}

using \text{13 le-bot by simp}

hence \text{222: injective ?w}

proof (rule injective-sup)
show injective (?v \cap - ?i)
   using 8 by (simp add: injective-inf-closed)
next
  show coreflexive (?e * (?v \cap - ?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 ?w
     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 \ast top)
     using 212 by simp
next
  show bijective (?e \ast top)
     using 14 212 by (smt assms(4) comp-inf.idempotent-bot-closed
core-complement minarc-arc minarc-bot-iff p-bot regular-closed-bot
   semiring.mult-not-zero symmetric-top-closed)
next
  show ?i \leq top * ?eT *?vT*
     using 19 by simp
next
  show ?v * ?eT * 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: \(?e \leq \cdots \leq \cdots h\)
by (metis inf.coboundedI1 inf.sup-monoid.add-commute minarc-below
order.trans p-dist-inf p-dist-sup sup.cobounded1)
thus \(?\text{thesis}\)
  using 1 2 by simp
qed
also have ... \(\leq \cdots\)
  using assms(20, 21) by auto
finally show \(?w \leq \cdots\)
  by simp
next
have 223: \(?i \leq (\cdots \leq \cdots)T \ast \cdots \ast \cdots\)
proof (rule boruvka-exchange-spanning-inv)
  show forest \(?v\)
    using 10 spanning-forest-def by blast
next
  show \(?v \ast \cdots \ast \cdots\) = \(?e \cdots \ast \cdots\)
    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 \(?i \leq \cdots \leq \cdots \ast \cdots \ast \cdots\)
    using 18 inf.sup-monoid.add-assoc by auto
next
  show arc \(?i\)
    using 212 by blast
next
  show arc \(?e\)
    using assms(18) minarc-arc minarc-bot-iff by auto
next
  show \(?v \leq \cdots\)
    using 10 spanning-forest-def by blast
next
  show \(?w \leq \cdots\)
proof –
  have 2231: \(?e \leq \cdots\)
    by (metis inf.boundedE minarc-below pp-dist-inf)
  have \(?w \leq \cdots \leq \cdots\)
    using inf-le1 sup-left-isotone by simp
  also have ... \(\leq \cdots\)
    using 2231 10 spanning-forest-def sup-least by blast
finally show \(?\text{thesis}\)
  by blast
qed
next
  show \(?e \leq \cdots\)
    by (metis inf.boundedE minarc-below pp-dist-inf)
next
  show components \(?v \leq \cdots\)
    by (simp add: 9)

84
have components $g \leq \text{forest-components } ?v$
  using 10 spanning-forest-def by auto
also have ... $\leq \text{forest-components } ?w$
proof (rule kruskal-exchange-forest-components-inv)
  next
  show injective (((?v \cap -?i) \sqcup ?e)
  using 222 by simp
  next
  show regular ?i
  using 15 by simp
  next
  show $?e \cdot \text{top} \cdot ?e = ?e$
  by (metis arc-top-arc minarc-arc-bot-iff semiring.mult-not-zero)
  next
  show $?i \leq \text{top} \cdot ?e^T \cdot ?e^*$
  using 19 by blast
  next
  show $?v \cdot ?e^T \cdot \text{top} = \text{bot}$
  using 13 by simp
  next
  show injective $?v$
  using 8 by simp
  next
  show $?i \leq ?v$
  by (simp add: le-infI1)
  next
  show $?i \leq (?v \cap -?i)^T * ?e^T \cdot \text{top}$
  using 223 by blast
qed
finally show components $g \leq \text{forest-components } ?w$
  by simp
next
  show regular ?w
  using 3 7 regular-conv-closed by simp
qed

next
  have 224: $?e \cap g \neq \text{bot}$
  using assms(18) inf.left-commute inf-bot-right minarc-meet-bot by fastforce
  have 225: sum ($?e \cap g$) $\leq$ sum ($?i \cap g$)
  proof (rule a-to-e-in-bigforest)
    show symmetric $g$
    using assms(1) borwka-inner-invariant-def borwka-outer-invariant-def
    by auto
  next
  show $j \neq \text{bot}$
  by (simp add: assms(19))
next
show \( f \leq g \)
by (simp add: assms(3))

next
show vector \( j \)
using assms(6) borwka-inner-invariant-def by blast

next
show forest \( h \)
by (simp add: assms(8))

next
show big-forest (forest-components \( h \)) \( d \)
by (simp add: assms(10))

next
show \( f \sqcup f^T = h \sqcup h^T \sqcup d \sqcup d^T \)
by (simp add: assms(14))

next
show \( \forall a \ b. \ bf-between-arcs \ a \ b \ (\ ?H \ ) \ d \land a \leq - \ ?H \sqcap - \ g \land b \leq d \longrightarrow \) sum \( (b \sqcap g) \leq \) sum \( (a \sqcap g) \)
by (simp add: assms(15))

next
show regular \( d \)
using assms(16) by auto

next
show \( ?e = ?e \)
by simp

next
show arc \( ?i \)
using 212 by blast

next
show \( \bf-between-arcs \ ?i \ ?e \ ?H \ (d \sqcup ?e) \)

proof
have \( d^T \times \ ?H \times ?e = \text{bot} \)
using assms(6, 7, 11, 12, 19) dT-He-eq-bot le-bot by blast

hence 251: \( d^T \times \ ?H \times ?e \leq (\ ?H \times d)^* \times \ ?H \times ?e \)
by simp

hence \( d^T \times \ ?H \times ?H \times ?e \leq (\ ?H \times d)^* \times \ ?H \times ?e \)
by (metis assms(8) forest-components-star star-star-decompose-9 mult-assoc)

hence \( d^T \times (\ ?H \times d)^* \times \ ?H \times ?e \leq (\ ?H \times d)^* \times \ ?H \times ?e \)
proof
have \( d^T \times \ ?H \times d \leq 1 \)
using assms(10) big-forest-def dTransHd-le-1 by blast

hence \( d^T \times \ ?H \times d \times (\ ?H \times d)^* \times \ ?H \times ?e \leq (\ ?H \times d)^* \times \ ?H \times ?e \)
by (metis mult-left-isotone star-circ-circ-mult star-involutive star-one)

hence \( d^T \times \ ?H \times ?e \sqcup d^T \times \ ?H \times d \times (\ ?H \times d)^* \times \ ?H \times ?e \leq (\ ?H \times d)^* \times \ ?H \times ?e \)
using 251 by simp
hence \( d^T \times (1 \sqcup ?H \times d \times (\ ?H \times d)^*) \times \ ?H \times ?e \leq (\ ?H \times d)^* \times \ ?H \times ?e \)
by (simp add: comp-associative comp-left-dist-sup)
semiring.distrib-right)
    thus \( \exists \)thesis
    by (simp add: star-left-unfold-equal)
qed
hence \( \exists \)H \( \forall \)d \( \exists \)H \( \exists \)e \( \exists \)e \leq \( \exists \)H \( \forall \)d \( \exists \)H \( \forall \)e
    by (simp add: mult-right-isotone mult-associative)
hence \( \exists \)H \( \exists \)d \( \exists \)H \( \exists \)e \( \exists \)e \leq \( \exists \)H \( \forall \)d \( \exists \)H \( \forall \)e
    by (smt star-slide mult-associative)
hence \( \exists \)H \( \exists \)d \( \exists \)H \( \exists \)e \( \exists \)e \leq \( \exists \)H \( \forall \)d \( \exists \)H \( \forall \)e
    by (metis assms(8) forest-components-star star.circ-decompose-9)
using star-slide by auto
hence \( \exists \)H \( \exists \)d \( \exists \)H \( \exists \)e \( \exists \)e \leq \( \exists \)H \( \forall \)d \( \exists \)H \( \forall \)e
    by (smt le-sup1 star.circ-loop-fixpoint sup.cobounded2 sup-commute
    mult-associative)
    hence \( \exists \)H \( \forall \)d \( \exists \)H \( \exists \)e \( \exists \)e \leq \( \exists \)H \( \forall \)d \( \exists \)H \( \forall \)e
    by (simp add: semiring.distrib-left semiring.distrib-right)
    hence \( \exists \)H \( \forall \)d \( \exists \)H \( \exists \)e \( \exists \)e \leq \( \exists \)H \( \forall \)d \( \exists \)H \( \forall \)e
    by (simp add: star-left-induct-mult mult-associative)
    hence 252: \( \exists \)H \( \forall \)d \( \exists \)H \( \exists \)e \( \exists \)e \leq \( \exists \)H \( \forall \)d \( \exists \)H \( \forall \)e
    by (smt mult-left-dist-sup star.circ-transitive-equal star-slide star-sup-1
    mult-associative)
    have \( \exists \)i \leq \( \exists \)top \( \exists \)e \( \exists \)F
    by auto
    hence \( \exists \)i \( \exists \)top \( \exists \)e \( \exists \)top \( \exists \)top
    by (simp add: conv-dist-comp conv-dist-inf mult-associative)
    hence \( \exists \)i \( \exists \)top \( \exists \)top \( \exists \)e \( \exists \)top
    by (simp add: comp-isotone by blast)
also have \( \exists \)e \( \exists \)top \( \exists \)top
    by (simp add: vector-mult-closed)
also have \( \exists \)e \( \exists \)top \( \exists \)top
    by (simp add: conv-dist-comp conv-star-commute)
also have \( \exists \)e \( \exists \)top
    by simp
also have \( \exists \)i \( \exists \)top \( \exists \)e \( \exists \)top
    by (simp add: assms(13))
also have \( \exists \)e \( \exists \)top \( \exists \)e \( \exists \)top
    by (simp add: 252 comp-isotone)
also have \( \exists \)e \( \exists \)top \( \exists \)e \( \exists \)top
    by (simp add: comp-isotone star-isotone)
finally have \( \exists \)i \( \exists \)top \( \exists \)top \( \exists \)e \( \exists \)top
    by blast
    thus \( \exists \)thesis
    using 212 assms(18) bf-between-arcs-def minarc-arc minarc-bot-iff by blast
qed
next
show \( \exists \)i \( \exists \)H \( \exists \)g
proof

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 \( ?\text{thesis} \)
  using 241 inf-greatest by blast
qed
next
show regular h
  using assms(20) by auto
qed
have \( ?v \cap ?e \cap -?i = \text{bot} \)
  using 14 by simp
hence \( \text{sum (} ?w \cap g \text{)} = \text{sum (} ?v \cap -?i \cap g \text{)} + \text{sum (} ?e \cap g \text{)} \)
  using sum-disjoint inf-commute inf-assoc by simp
also have \( \ldots \leq \text{sum (} ?v \cap -?i \cap g \text{)} + \text{sum (} ?i \cap g \text{)} \)
  using 224 225 sum-plus-right-isotone by simp
also have \( \ldots = \text{sum ((} ?v \cap -?i \cup ?i) \cap g \text{)} \)
  using sum-disjoint inf-le2 pseudo-complement by simp
also have \( \ldots = \text{sum ((} ?v \cup ?i) \cap (\neg ?i \cup ?i) \cap g \text{)} \)
  by (simp add: sup-inf-distrib2)
also have \( \ldots = \text{sum ((} ?v \cup ?i) \cap g \text{)} \)
  using 15 by (metis inf-top-right stone)
also have \( \ldots = \text{sum (} ?v \cap g \text{)} \)
  by simp
thus \( \forall u . \text{spanning-forest } u \rightarrow \text{sum (} ?w \cap g \text{)} \leq \text{sum (} u \cap g \text{)} \)
  using 2 11 minimum-spanning-forest-def by auto
qed
next
have \( ?f \leq f \cap f^T \cap ?e \)
  by (smt cone-dist-inf inf-le1 sup-left-isotone sup-mono inf.order-lesseq-imp)
also have \( \ldots \leq (\text{sup-left-isotone by simp}) \)
  using 20 sup-left-isotone by simp
also have \( \ldots \leq (\text{sup-coclosed sup-inf-distrib2}) \)
  by (metis inf.cobounded1 sup-inf-distrib2)
also have \( \ldots = ?w \cup (\text{sup-assoc sup-commute}) \)
  by simp
also have \( \ldots \leq \text{sup-right-isotone inf-associative sup-right-isotone by simp} \)
also have \( \ldots \leq \text{sup-right-isotone inf-associative sup-right-isotone by simp} \)
  using conv-complement conv-dist-inf conv-dist-sup sup-right-isotone by simp
  by simp
finally show \( \text{if} \leq ?w \cup ?w^T \)
  by simp
qed
thus \( ?\text{thesis} \) by auto
qed

88
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 ∩ − path f h j g △ (f ∩ − selected-edge h j g) ⊔ 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 ?d′ = d △ ?e
let ?j′ = j ⊓ − ?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 ⊓ − ?eT)
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 ⊓ − ?eT)T ≤ ?p
by (simp add: mult-right-isotone star.left-plus-below-circ star-plus mult-assoc)
show ?e ≤ ?p
by (meson mult-left-isotone order.trans star-outer-increasing top.extremum)
show ?p * (f ⊓ − ?eT)T ≤ − ?e
proof −
have ?p * (f ⊓ − ?eT)T ≤ ?p * fT
by (simp add: conv-dist-inf mult-right-isotone)
also have ... ≤ top * ?e * (f)T* * fT
using conv-dist-inf star-isotone comp-isotone by simp
also have ... ≤ − ?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 (\( ?p^T \ast ?p \cap (f \cap - ?e^T)^T \ast (f \cap - ?e^T) \))

proof -
  have (\( ?p^T \ast ?p \cap (f \cap - ?e^T)^T \ast (f \cap - ?e^T) \)) \( \leq \) \( ?p^T \ast ?p \cap f^T \ast f \)
  using conv-dist-inf inf.sup-right-isotone mult-isotone by simp
  also have \( ... \leq (\text{top} \ast ?e \ast f^{T\ast})^T \ast (\text{top} \ast ?e \ast f^{T\ast}) \cap f^T \ast f \)
  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 \( ... \leq 1 \)
  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 \( \text{thesis} \)
  by simp
qed

next

show acyclic \( ?f' \)
proof (rule kruskal-acyclic-inv)
show acyclic (\( f \cap - ?e^T \))
proof -
  have f-intersect-below: (\( f \cap - ?e^T \)) \( \leq f \) by simp
  have acyclic f
    by (meson assms(1) boruvka-inner-invariant-def boruvka-outer-invariant-def)
  thus \( \text{thesis} \)
    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 (\( f \cap - ?e^T \cap ?p \))\( ^T \ast (f \cap - ?e^T)^T \ast ?e = \text{bot} \)
proof -
  have \( ?e \leq - (f^{T\ast} \ast f^{\ast}) \)
    by (simp add: assms(3))
  hence \( ?e \ast \text{top} \ast ?e \leq - (f^{T\ast} \ast f^{\ast}) \)
    by (metis arc-top-arc minarc-arc minarc-bot-iff semiring.mult-not-zero)
  hence \( ?e^T \ast \text{top} \ast ?e^T \leq - (f^{T\ast} \ast f^{\ast})^T \)
    by (metis comp-associative conv-complement conv-dist-comp comp-isotone symmetric-top-closed)
  hence \( ?e^T \ast \text{top} \ast ?e^T \leq - (f^{T\ast} \ast f^{\ast}) \)
    by (simp add: conv-dist-comp conv-star-commute)
  hence \( ?e \ast (f^{T\ast} \ast f^{\ast}) \ast ?e \leq \text{bot} \)
    using triple-schroeder-p by auto
  hence \( 1: ?e \ast f^{T\ast} \ast f^{\ast} \ast ?e \leq \text{bot} \)
    using mult-assoc by auto
  have 2: (\( f \cap - ?e^T \))\( ^T \ast f^{\ast} \leq f^T \ast f^{\ast} \)
    by (simp add: conv-dist-inf star-isotone)
  have (\( f \cap - ?e^T \cap ?p \))\( ^T \ast (f \cap - ?e^T)^T \ast ?e \leq (f \cap ?p)^T \ast (f \cap -
\( (e^T)^* \leq e \)

by \((\text{simp add: comp-isotone conv-dist-inf inf.orderI inf.sup-monoid.add-assoc})\)
also have ... \( \leq (f \cap \gamma p)^T \cdot f^* \cdot \gamma e \)
by \((\text{simp add: comp-isotone star-isotone})\)
also have ... \( \leq (f \cap \gamma \text{top} \cdot \gamma e \cdot (f^T)^*)^T \cdot f^* \cdot \gamma e \)
using 2 by \((\text{metis comp-inf.comp-isotone comp-inf.coreflexive-transitive comp-isotone conv-isotone inf.orderI sup-monoid.add-assoc})\)
also have ... \( = (f \cap \gamma \text{top} \cdot \gamma e \cdot (f^T)^*) \cdot \gamma e \)
by \((\text{smt covector-comp-inf-1 covector-mult-closed eq-iff})\)
also have ... \( \leq \text{top} \cdot \gamma e \cdot (f^T)^* \cdot \gamma e \)
using \((\text{top-left-mult-increasing mult-assoc by auto})\)
also have ... \( = \text{top} \cdot \gamma e \cdot (f^T)^* \cdot \gamma e \)
by \((\text{simp})\)
also have ... \( \leq \text{bot} \)
using \((\text{covector-bot-closed le-bot mult-assoc by fastforce})\)
finally show \( ?\text{thesis} \)
using \((\text{le-bot by auto})\)
qed

next
show \( ?e \cdot (f \cap -e)^T \cdot \gamma e = \text{bot} \)
proof
have 1: \( ?e \leq -\gamma F \)
by \((\text{meson assms(3)})\)
have 2: injective \( f \)
by \((\text{meson assms(1) borwka-inner-invariant-def borwka-outer-invariant-def})\)
have 3: equivalence \( ?F \)
using 2 \((\text{forest-components-equivalence by simp})\)
hence 4: \( ?e^T = \gamma e \cdot (f^T)^* \cdot \gamma e \)
also have ... \( \leq -\gamma F \)
using 1 3 \((\text{conv-isotone conv-complement calculation by fastforce})\)

finally have 5: \( ?e \cdot ?F \cdot ?e = \text{bot} \)
using 4 by \((\text{smt triple-schroeder-p le-bot pp-total regular-closed-top vector-top-closed})\)
have \( (f \cap -e^T)^* \leq f^* \)
by \((\text{simp add: star-isotone})\)
hence \( ?e \cdot (f \cap -e^T)^* \cdot ?e \leq ?e \cdot f^* \cdot ?e \)
using \((\text{mult-left-isotone mult-right-isotone by blast})\)
also have ... \( \leq ?e \cdot ?F \cdot ?e \)
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 ⊓ -eT) ≤ - e
proof -
have 1: ?e ≤ - ?F
  by (simp add: assms(3))
have f ⊓ - ?eT ≤ f
  by simp
hence forest-components (f ⊓ - ?eT) ≤ ?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 ⊓ - ?eT ⊓ - ?p) ≤ - - g
  by (meson assms(1) boruvka-inner-invariant-def
boruvka-outer-invariant-def inf.coboundedII)
have 2: (f ⊓ - ?eT ⊓ ?p)T ≤ - - g
proof -
  have (f ⊓ - ?eT ⊓ ?p)T ≤ fT
    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 ⊔ wT
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) boruva-inner-invariant-def boruwa-outer-invariant-def  
by auto  
  next  
  show \( f \leq -g \)  
  using assms(1) boruva-inner-invariant-def boruwa-outer-invariant-def  
by auto  
  next  
  show regular f  
  using assms(1) boruva-inner-invariant-def boruwa-outer-invariant-def  
by auto  
  next  
  show (\( \exists w . \) minimum-spanning-forest w g \( \land f \leq w \sqcup w^T \))  
  using assms(1) boruva-inner-invariant-def boruwa-outer-invariant-def  
by auto  
  next  
  show vector j  
  using assms(1) boruva-inner-invariant-def by blast  
next  
  show regular j  
  using assms(1) boruva-inner-invariant-def by blast  
next  
  show forest h  
  using assms(1) boruva-inner-invariant-def by blast  
next  
  show forest-components h \( \leq \) forest-components f  
  using assms(1) boruva-inner-invariant-def by blast  
next  
  show big-forest (forest-components h) d  
  using assms(1) boruva-inner-invariant-def by blast  
next  
  show d * top \( \leq -j \)  
  using assms(1) boruva-inner-invariant-def by blast  
next  
  show forest-components h * j = j  
  using assms(1) boruva-inner-invariant-def by blast  
next  
  show forest-components f = (forest-components h * (d \( \sqcup d^T \)))^*  
  forest-components h  
  using assms(1) boruva-inner-invariant-def by blast  
next  
  show f \( \sqcup f^T = h \sqcup h^T \sqcup d \sqcup d^T \)  
  using assms(1) boruva-inner-invariant-def by blast  
next  
  show (\( \forall a b . \) bf-between-arcs a b (forest-components h) d \( \land a \leq -(forest-components h) \sqcap -g \land b \leq d \longrightarrow \) sum(b \( \sqcap g \)) \( \leq \) sum(a \( \sqcap g \)))  
  using assms(1) boruva-inner-invariant-def by blast
next
  show regular d
  using assms(1) borueva-inner-invariant-def by blast
next
  show selected-edge h j g ≤ - forest-components f
  by (simp add: assms(3))
next
  show selected-edge h j g ≠ bot
  by (simp add: assms(4))
next
  show j ≠ bot
  by (simp add: assms(2))
next
  show regular h
  using assms(1) borueva-inner-invariant-def borueva-outer-invariant-def
  by auto
next
  show h ≤ -- g
  using assms(1) borueva-inner-invariant-def borueva-outer-invariant-def
  by auto
qed
qed
qed

lemma second-inner-invariant-when-e-not-bot:
  assumes borueva-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 borueva-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 ⊓ -?e
  show borueva-inner-invariant ?j' ?f' h g ?d'
proof (unfold borueva-inner-invariant-def, intro conjI)
  have 1: borueva-outer-invariant ?f' g
    using assms(1, 2, 3, 4) borueva-outer-invariant-when-e-not-bot
    by blast
  show borueva-outer-invariant ?f' g

using assms(1, 2, 3, 4) boruvka-outer-invariant-when-e-not-bot by blast
show g ≠ bot using assms(1) boruvka-inner-invariant-def by force
show vector ?j′ using assms(1, 2) boruvka-inner-invariant-def component-is-vector
vector-complement-closed vector-inf-closed by simp
show regular ?j′ using assms(1) boruvka-inner-invariant-def by auto
show boruvka-outer-invariant h g by (meson assms(1) boruvka-inner-invariant-def)
show injective h by (meson assms(1) boruvka-inner-invariant-def)
show pd-kleene-allegory-class.acyclic h by (meson assms(1) boruvka-inner-invariant-def)
show ?H ≤ forest-components ?f′ proof –
  have 2: ?F ≤ forest-components ?f′ proof (rule components-disj-increasing)
    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[1]
next
  show regular ?e using assms(1) boruvka-outer-invariant-def boruvka-outer-invariant-def
minarc-regular regular-closed-star regular-conv-closed regular-mult-closed by auto[1]
next
  show injective ?f′ using 1 boruvka-outer-invariant-def by blast
next
  show injective f using assms(1) boruvka-inner-invariant-def boruvka-outer-invariant-def
by blast
qed
thus thesis using assms(1) boruvka-inner-invariant-def dual-order.trans by blast
qed
show big-forest ?H ?d′ using assms(1, 2, 3, 4) big-forest-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 auto
hence 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
hence ?e * top ≤ ?c
by (metis component-is-vector mult-left-isotone)
also have ... ≤ − j Ω ?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 ?H * ?j = ?j'
using fc-j-eq-j-inv assms(1) boruvka-inner-invariant-def by blast
next
show forest-components ?f' = (?H * (?d' ⊓ ?d'^T)) * ?H
proof −
have forest-components ?f' = (f ⊓ f^T ⊓ ?e ⊓ ?e^T)^*
proof (rule simplify-forest-components-f)
show regular ?p
using assms(1) boruvka-inner-invariant-def boruvka-outer-invariant-def
minarc-regular regular-conv-closed-Star regular-conv-closed regular-mult-closed by auto
next
show regular ?e
using minarc-regular by auto
next
show injective ?f'
using assms(1, 2, 3, 4) boruvka-outer-invariant-def
boruvka-outer-invariant-when-e-not-bot by blast
next
show injective f
using assms(1) boruvka-inner-invariant-def boruvka-outer-invariant-def
by blast
qed
also have ... = (h ⊓ h^T ⊓ d ⊓ d^T ⊓ ?e ⊓ ?e^T)^*
using assms(1) boruwna-inner-invariant-def by simp
also have ... = (\{ p \} \cup h^T) \cup ?d' \cup ?d'^T)
  by (smt conv-dist-sup sup-monoid.add-assoc sup-monoid.add-commute)
also have ... = ((h \cup h^T)^* \circ ((\{d' \cup \{d'^T\})^* \circ (h \cup h^T)^*)
  by (metis star.circ-sup-3 sup-assoc)
finally show \$thesis
  using assms(1) boruwna-inner-invariant-def forest-components-wcc by simp
next
  show \$thesis
    using assms(1) boruwna-inner-invariant-def boruwna-outer-invariant-def
    minarc-regular regular-closed-star regular-conv-closed regular-mult-closed by auto

next
  show \$thesis
    using minarc-regular by blast
qed
also have ... = h \cup h^T \cup d \cup d^T \cup ?e \cup ?e^T
  using assms(1) boruwna-inner-invariant-def by auto
finally show \$thesis
  by (smt conv-dist-sup sup.left-commute sup-commute)
qed
next
  show \\forall \ a \ b \ . \ bfr-between-arcs a b \ ?H \ ?d' \land \ a \leq \ - \ ?H \ \top \ g \ \leq \ ?d' \ \top \ \sum (b \ \{ g \} \leq \ \sum (a \ \{ g \)
  proof (intro allI, rule impl, unfold bfr-between-arcs-def)
  fix a b
  assume 1: (arc a \land arc b \land a^T \ast top \leq (\{H \ast \{d'\}^* \ast \{H \ast b \ast top\}) \land \ a \leq \ - \ ?H \ \top \ g \ \leq \ ?d'
  thus \sum (b \ \{ g \} \leq \ \sum (a \ \{ 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 \ \{ g \} \leq \ \sum (a \ \{ g \)

97
proof (rule a-to-e-in-bigforest)
  show symmetric g
    using assms(1) boruvka-inner-invariant-def
    boruvka-outer-invariant-def by auto
next
  show j ≠ bot
    by (simp add: assms(2))
next
  show f ≤ 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 big-forest (forest-components h) d
    using assms(1) boruvka-inner-invariant-def by blast
next
  show f ⊔ fT = h ⊔ hT ⊔ d ⊔ dT
    using assms(1) boruvka-inner-invariant-def by blast
next
  show ∀ a b. bf-between-arcs 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 bf-between-arcs a b ?H ?d′
    using 1 bf-between-arcs-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
  hence b-below-d: \( b \leq d \)
    using 1 by (metis assms(4) different-arc-in-sup-arc minarc-arc
      minarc-bot-iff)
    thus \( ?thesis \)

  proof (cases \( ?e \leq d \))
    case True
    hence bf-between-arcs a b ?H d \( \land \) \( b \leq d \)
      using 1 bf-between-arcs-def sup.absorb1 by auto
    thus \( ?thesis \)
      using 1 assms(1) boruoka-inner-invariant-def by blast

next

  case e-not-less-than-d: False
  have 71: equivalence ?H
    using assms(1) fch-equivalence boruoka-inner-invariant-def by auto
  hence 72: \( \text{bf-between-arcs a b ?H d} \land \text{bf-between-arcs a b ?H d} \lor \)
    (\( \text{bf-between-arcs a ?e ?H d} \land \text{bf-between-arcs ?e b ?H d} \))
    proof (rule big-forest-path-split-disj)
      show arc ?e
        using assms(4) minarc-arc minarc-bot-iff by blast
      next
      show regular a \( \land \) regular b \( \land \) regular ?e \( \land \) regular d \( \land \) regular ?H
        using assms(1) 1 boruoka-inner-invariant-def
        boruoka-outer-invariant-def arc-regular minarc-regular regular-closed-star
        regular-cone-closed regular-mult-closed by auto
    qed
    thus \( ?thesis \)
    proof (cases \( \text{bf-between-arcs a b ?H d} \))
      case True
      have bf-between-arcs a b ?H d \( \land \) \( b \leq d \)
        using 1 by (metis assms(4) True b-not-equal-e minarc-arc
          minarc-bot-iff different-arc-in-sup-arc)
      thus \( ?thesis \)
        using 1 assms(1) b-below-d boruoka-inner-invariant-def by auto

next

  case False
  have 73: \( \text{bf-between-arcs a ?e ?H d} \land \text{bf-between-arcs ?e b ?H d} \)
    using 1 72 False bf-between-arcs-def by blast
  have 74: \( ?e \leq - - g \)
    by (metis inf.boundedE minarc-below pp-dist-inf)
  have \( ?e \leq - ?H \)
    by (meson assms(1, 3) boruoka-inner-invariant-def dual-order.trans
      p-antitone-iff)
  hence \( ?e \leq - - g \)
    using 74 by simp
  hence 75: \( \text{sum (b \land g)} \leq \text{sum (？e \land g)} \)
    using assms(1) b-below-d 73 boruoka-inner-invariant-def by blast
have 76: bf-between-arcs a ?e ?H ?d′
using 73 by (meson big-forest-path-split-disj assms(1)
bf-between-arcs-def boruvka-inner-invariant-def boruvka-outer-invariant-def
fch-equivalence arc-regular regular-closed-star regular-conv-closed
regular-mult-closed)

have 77: \( \sum (\text{?e } \cap \text{g}) \leq \sum (\text{a } \cap \text{g}) \)

proof (rule a-to-e-in-bigforest)
show symmetric g
using assms(1) boruvka-inner-invariant-def

boruvka-outer-invariant-def by auto

next
show \( j \neq \text{bot} \)
by (simp add: assms(2))

next
show \( f \leq \text{g} \)
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 big-forest \( (\text{forest-components \( h \)} \text{ d}) \)
using assms(1) boruvka-inner-invariant-def by blast

next
show \( f \cup f^T = h \cup h^T \cup d \cup d^T \)
using assms(1) boruvka-inner-invariant-def by blast

next
show \( \forall a \ b. \text{bf-between-arcs a b (}\text{?H} \text{ d) a } \leq \text{\(-}\text{?H } \cap \text{-} \text{g) a} \leq \text{\(-}\text{g) a} \leq \text{\(-}\text{d) \( a \cap \text{g) \leq \sum (a } \cap \text{g) \)} \)
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 bf-between-arcs a ?e ?H ?d′
by (simp add: 76)

next
show \( a \leq \text{\(-}\text{?H } \cap \text{-} \text{g) \)}
using 1 by simp

next
show regular \( h \)

using \( \text{assms}(1) \) \( \text{boruvka-inner-invariant-def} \)

\( \text{boruvka-outer-invariant-def} \) by \( \text{auto} \)

qed

thus \( \text{thesis} \)

using \( 75 \) order_trans by \( \text{blast} \)

qed

qed

qed

next

show regular \( \?d' \)

using \( \text{assms}(1) \) \( \text{boruvka-inner-invariant-def} \) \( \text{minarc-regular} \) by \( \text{auto} \)

qed

qed

lemma \( \text{second-inner-invariant-when-e-bot} \):  
assumes \( \text{selected-edge} \ h \ j \ g = \text{bot} \)

and \( \text{selected-edge} \ h \ j \ g \leq \text{-forest-components} \ f \)

and \( \text{boruvka-inner-invariant} \ j \ f \ h \ g \ d \)

shows \( \text{boruvka-inner-invariant} \)

\((j \cap - \text{choose-component} (\text{forest-components} h) j) \)

\((f \cap - \text{selected-edge} h j g^T \cap - \text{path} f h j g \sqcup) \)

\((f \cap - \text{selected-edge} h j g^T \cap \text{path} f h j g)^T \sqcup \text{selected-edge} h j g) \)

\( h \ g \ (d \sqcup \text{selected-edge} h j g) \)

proof –

let \( \?c = \text{choose-component (forest-components} h) j \)

let \( \?p = \text{path} f h j g \)

let \( \?F = \text{forest-components} f \)

let \( \?H = \text{forest-components} h \)

let \( \?e = \text{selected-edge} h j g \)

let \( \?f' = f \cap -?e^T \cap -?p \sqcup (f \cap -?e^T \cap ?p)^T \sqcup \?e \)

let \( \?d' = d \sqcup \?e \)

let \( \?j' = j \cap -?c \)

show \( \text{boruvka-inner-invariant} \ ?j' \ ?f' \ h \ g \ ?d' \)

proof (unfold \( \text{boruvka-inner-invariant-def} \), intro conjI)

next

show \( \text{boruvka-outer-invariant} \ ?f' \ g \)

using \( \text{assms}(1, 3) \) \( \text{boruvka-inner-invariant-def} \) by \( \text{auto} \)

next

show \( g \neq \text{bot} \)

using \( \text{assms}(3) \) \( \text{boruvka-inner-invariant-def} \) by \( \text{blast} \)

next

show \( \text{vector} \ ?j' \)

by (metis \( \text{assms}(3) \) \( \text{boruvka-inner-invariant-def} \) \( \text{component-is-vector} \)

\( \text{vector-complement-closed} \) \( \text{vector-inf-closed} \))

next

show \( \text{regular} \ ?j' \)

101
using assms(3) boruvka-inner-invariant-def by auto
next
  show boruvka-outer-invariant h g
  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 ≤ forest-components ?f'
  using assms(1, 3) boruvka-inner-invariant-def by auto
next
  show big-forest ?H ?d'
  using assms(1, 3) boruvka-inner-invariant-def by auto
next
  show ?d' * top ≤ −?j'
    by (metis assms(1, 3) boruvka-inner-invariant-def order.trans p-antitone-inf sup-monoid.add-0-right)
next
  show ?H * ?j' = ?j'
    using assms(3) fc-j-eq-j-inv boruvka-inner-invariant-def by blast
next
  show forest-components ?f' = (?H * (?d' ∪ ?d'T)) * ?H
    using assms(1, 3) boruvka-inner-invariant-def by auto
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. bf-between-arcs 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
next
  show regular ?d'
    using assms(1, 3) boruvka-inner-invariant-def by auto
qed
qed

4.4 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

\( j := \text{top} \);
\( h := f \);
\( d := \text{bot} \);

WHILE \( j \neq \text{bot} \)

INV { boruvka-inner-invariant \( j \) \( f \) \( h \) \( g \) \( d \) }

DO

\( c := \text{choose-component} (\text{forest-components} \ h) \ j \);
\( e := \text{minarc} (c - c^T \cap g) \);

\( \text{IF} \ e \leq -(\text{forest-components} \ f) \ THEN \)

\( f := f \cap -e^T \);
\( f := (f \cap -(\text{top} * e * f^{T*})) \sqcup (f \cap \text{top} * e * f^{T*})^T \sqcup e \);
\( d := d \sqcup e \)

ELSE

\( \text{SKIP} \)

\( \text{FI} \)

\( j := j \cap -c \)

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 \( f \) \( g \) \& \(- ?F \cap g \neq \text{bot} \)

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 \( g \neq \text{bot} \)

using \( f \) by auto

next

show surjective top

by simp

next

show regular top

by simp

next

show boruvka-outer-invariant \( f \) \( g \)

using \( f \) by auto

next

show injective \( f \)
begin

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 ?F ≤ ?F
by simp
next
show big-forest ?F bot
by (simp add: 2 big-forest-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 * (bot ⊔ botT)) * ?F
by (metis mult-right-zero semiring.mult-zero-left star.circ-loop-fixpoint sup-commute sup-monoid.add-0-right symmetric-bot-closed)
next
show f ⊔ fT = f ⊔ fT ⊔ bot ⊔ botT
by simp
next
show ∀ a b. bf-between-arcs a b ?F bot ∧ a ≤ − ?F ⊓ − − g ∧ b ≤ bot → sum (b ⊓ g) ≤ sum (a ⊓ g)
by (metis (full-types) bf-between-arcs-def bot-unique mult-left-zero mult-right-zero top.extremum)
next
show regular bot
by auto
qed
next
fix f j h d
let ?e = 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 ?d' = d ⊓ ?e
let ?j' = j ⊓ − ?e
assume 1: boruvka-inner-invariant j f h g d ∧ j ≠ bot
proof (intro conjI)
show ?e ≤ − ?F → boruvka-inner-invariant ?j' ?f' h g ?d'
proof (cases ?e = bot)
case True

104
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 \( ?e \leq - ?F \rightarrow \text{boruwka-inner-invariant} \ ?j' \ f \ h \ g \ d \)
proof (rule impI, unfold boruwka-inner-invariant-def, intro conjI)
  show "boruwka-outer-invariant" f g
    using 1 boruwka-inner-invariant-def by blast
next
  show "g \neq \text{bot}" using 1 boruwka-inner-invariant-def by blast
next
  show "vector ?j'"
    using 1 boruwka-inner-invariant-def component-is-vector
vector-complement-closed vector-inf-closed by auto
next
  show regular ?j'
    using 1 boruwka-inner-invariant-def by auto
next
  show "boruwka-outer-invariant h g"
    using 1 boruwka-inner-invariant-def by auto
next
  show injective h
    using 1 boruwka-inner-invariant-def by blast
next
  show pd-kleene-allegory-class.acyclic h
    using 1 boruwka-inner-invariant-def by blast
next
  show \( ?H \leq ?F \)
    using 1 boruwka-inner-invariant-def by blast
next
  show big-forest ?H d
    using 1 boruwka-inner-invariant-def by blast
next
  show \( d \ast \text{top} \leq - ?j' \)
    using 1 by (meson boruwka-inner-invariant-def dual-order.trans
p-antitone-inf)
next
  show \( ?H \ast ?j' = ?j' \)
    using 1 fc-j-eq-j-inv boruwka-inner-invariant-def by blast
next
  show \( ?F = (?H \ast (d \sqcup d^T)) \ast ?H \)
    using 1 boruwka-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 \( \neg e \leq -F \implies \forall a \ b, \textbf{bf-between-arcs} \ a \ b \ \leq -F \land -g \)
\( \land b \leq d \implies \text{sum}(b \cap g) \leq \text{sum}(a \cap g) \)
using 1 boruvka-inner-invariant-def by blast
next
show \( \neg e \leq -F \implies \text{regular} \ d \)
using 1 boruvka-inner-invariant-def by blast
qed
qed
next
fix f h d j
assume boruvka-inner-invariant j 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 \( \land -\text{forest-components} \ f \land g = \text{bot} \)
hence 2:spanning-forest f g
proof (unfold spanning-forest-def, intro conj)
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 \( \leq \text{forest-components} \ f \)
proof –
let \( ?F = \text{forest-components} \ f \)
have \( \neg ?F \land g \leq \text{bot} \)
by (simp add: 1)
hence \( -g \leq \text{bot} \land -?F \)
using 1 shunting-p p-antitone pseudo-complement by auto
hence \( -g \leq ?F \)
using 1 boruvka-outer-invariant-def pp-dist-comp pp-dist-star
regular-conv-closed by auto
hence \((\neg g)^* \leq ?F^* \)
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 \( \land f \leq w \downarrow w^T \)
using \(\text{boruvka-outer-invariant-def}\) by \text{blast}

hence \(w = w \cap -g\)

by (simp add: \text{inf.absorb1} \text{minimum-spanning-forest-def} \text{spanning-forest-def})

also have \(\ldots \leq w \cap \) components \(g\)

by (metis \text{inf.sup-right-isotone} \text{star.circ-increasing})

also have \(\ldots \leq w \cap f^T \ast f^*\)

using 2 \text{spanning-forest-def} \text{inf.sup-right-isotone} by simp

also have \(\ldots \leq f \cup f^T\)

proof (rule cancel-separate-6[where \(z=w\) and \(y=w^T\)])

show injective \(w\)

using 3 \text{minimum-spanning-forest-def} \text{spanning-forest-def} by simp

next

show \(f^T \leq w^T \cup w\)

using 3 by (metis \text{conv-dist-inf} \text{conv-dist-sup} \text{conv-involutive} \text{inf.cobounded2} \text{inf.orderE})

next

show \(f \leq w^T \cup w\)

using 3 by (simp add: \text{sup-commute})

next

show injective \(w\)

using 3 \text{minimum-spanning-forest-def} \text{spanning-forest-def} by simp

next

show \(w \cap w^T \ast = \text{bot}\)

using 3 by (metis \text{acyclic-star-below-complement} \text{comp-inf} \text{mult-right-isotone} \text{inf-p le-bot} \text{minimum-spanning-forest-def} \text{spanning-forest-def})

qed

finally have 4: \(w \leq f \cup f^T\)

by simp

have \(\text{sum} (f \cap g) = \text{sum} ((w \cup w^T) \cap (f \cap g))\)

using 3 by (metis \text{inf.absorb2} \text{inf.assoc})

also have \(\ldots = \text{sum} (w \cap (f \cap g)) + \text{sum} (w^T \cap (f \cap g))\)

using 3 \text{inf.commute} \text{acyclic-asymmetric} \text{sum-disjoint}

\text{minimum-spanning-forest-def} \text{spanning-forest-def} by simp

also have \(\ldots = \text{sum} (w \cap (f \cap g)) + \text{sum} (w^T \cap (f \cap g^T))\)

by (metis \text{conv-dist-inf} \text{conv-involutive} \text{sum-conv})

also have \(\ldots = \text{sum} (f \cap (w \cap g)) + \text{sum} (f^T \cap (w \cap g))\)

proof –

have 51: \(f^T \cap (w \cap g) = f^T \cap (w \cap g^T)\)

using 1 \text{boruvka-outer-invariant-def} by \text{auto}

have 52: \(f \cap (w \cap g) = w \cap (f \cap g)\)

by (simp add: \text{inf.left-commute})

thus \(\text{thesis}\)

using 51 52 \text{abel-semigroup.left-commute} \text{inf.abel-semigroup-axioms} by \text{fastforce}

qed

also have \(\ldots = \text{sum} ((f \cup f^T) \cap (w \cap g))\)

using 2 \text{acyclic-asymmetric} \text{inf.sup-monoid.add-commute} \text{sum-disjoint}

\text{spanning-forest-def} by simp

also have \(\ldots = \text{sum} (w \cap g)\)
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 přírodo-
 vědecké společnosti, 3(3):37–58, 1926.


[3] W. Guttmann. Relation-algebraic verification of Prim’s minimum span-
ning tree algorithm. In A. Sampaio and F. Wang, editors, Theoretical
Aspects of Computing – ICTAC 2016, volume 9965 of Lecture Notes in


[5] W. Guttmann. An algebraic framework for minimum spanning tree prob-

relation algebras. Journal of Logical and Algebraic Methods in Program-

spanning tree problem – Translation of both the 1926 papers, comments,

https://doi.org/10.26021/10196.