An Axiomatic Characterization of the Single-Source Shortest Path Problem

By Christine Rizkallah

April 20, 2020

Abstract

This theory is split into two sections. In the first section, we give a formal proof that a well-known axiomatic characterization of the single-source shortest path problem is correct. Namely, we prove that in a directed graph $G = (V, E)$ with a non-negative cost function on the edges the single-source shortest path function $\mu : V \rightarrow \mathbb{R} \cup \{\infty\}$ is the only function that satisfies a set of four axioms. The first axiom states that the distance from the source vertex $s$ to itself should be equal to zero. The second states that the distance from $s$ to a vertex $v \in V$ should be infinity if and only if there is no path from $s$ to $v$. The third axiom is called triangle inequality and states that if there is a path from $s$ to $v$, and an edge $(u,v) \in E$, the distance from $s$ to $v$ is less than or equal to the distance from $s$ to $u$ plus the cost of $(u,v)$. The last axiom is called justification, it states that for every vertex $v$ other than $s$, if there is a path $p$ from $s$ to $v$ in $G$, then there is a predecessor edge $(u,v)$ on $p$ such that the distance from $s$ to $v$ is equal to the distance from $s$ to $u$ plus the cost of $(u,v)$.

In the second section, we give a formal proof of the correctness of an axiomatic characterization of the single-source shortest path problem for directed graphs with general cost functions $c : E \rightarrow \mathbb{R}$. The axioms here are more involved because we have to account for potential negative cycles in the graph. The axioms are summarized in the three isabelle locales.

Contents

1 Shortest Path (with non-negative edge costs) 2
2 Shortest Path (with general edge costs) 8

theory ShortestPath

imports

  Graph-Theory, Graph-Theory

begin
1 Shortest Path (with non-negative edge costs)

The following theory is used in the verification of a certifying algorithm’s checker for shortest path. For more information see [1].

locale basic-sp =
  fin-digraph +
fixes dist :: 'a ⇒ ereal
fixes c :: 'b ⇒ real
fixes s :: 'a
assumes general-source-val: dist s ≤ 0
assumes trian:
  ∀ e. e ∈ arcs G ⇒
  dist (head G e) ≤ dist (tail G e) + c e

locale basic-just-sp =
  basic-sp +
fixes num :: 'a ⇒ enat
assumes just:
  ∀ v. [v ∈ verts G; v ≠ s; num v ≠ ∞] ⇒
  ∃ e ∈ arcs G. v = head G e ∧
  dist v = dist (tail G e) + c e ∧
  num v = num (tail G e) + (enat 1)

locale shortest-path-pos-cost =
  basic-just-sp +
assumes s-in-G: s ∈ verts G
assumes tail-val: dist s = 0
assumes no-path: ∃ v. v ∈ verts G ⇒ dist v = ∞ ↔ num v = ∞
assumes pos-cost: ∀ e. e ∈ arcs G ⇒ 0 ≤ c e

locale basic-just-sp-pred =
  basic-sp +
fixes num :: 'a ⇒ enat
fixes pred :: 'a ⇒ 'b option
assumes just:
  ∀ v. [v ∈ verts G; v ≠ s; num v ≠ ∞] ⇒
  ∃ e ∈ arcs G.
  e = the (pred v) ∧
  v = head G e ∧
  dist v = dist (tail G e) + c e ∧
  num v = num (tail G e) + (enat 1)

sublocale basic-just-sp-pred ⊆ basic-just-sp
using basic-just-sp-pred-axioms
unfolding basic-just-sp-pred-def
basic-just-sp-pred-axioms-def
by unfold-locales (blast)

locale shortest-path-pos-cost-pred =
assumes $s \in \mathrm{verts} \ G$
assumes tail-val: \( \text{dist} \ s = 0 \)
assumes no-path: $\forall v. \ v \in \mathrm{verts} \ G \implies \text{dist} \ v = \infty \iff \text{num} \ v = \infty$
assumes pos-cost: $\forall e. \ e \in \mathrm{arcs} \ G \implies 0 \leq c \ e$

sublocale shortest-path-pos-cost-pred \subseteq shortest-path-pos-cost
using shortest-path-pos-cost-pred-axioms by unfold-locales
(auto simp: shortest-path-pos-cost-pred-def shortest-path-pos-cost-pred-axioms-def)

lemma tail-value-helper:
assumes hd p = last p
assumes distinct p
assumes $p \neq []$
shows $p = [\text{hd} \ p]$
by (metis assms distinct simp (2) list.sel(1) neq-Nil-conv last-ConsR last-in-set)

lemma (in basic-sp) dist-le-cost:
fixes $v :: 'a$
fixes $p :: 'b \ \mathrm{list}$
assumes awalk s p v
shows $\text{dist} \ v \leq \text{awalk-cost} \ c \ p$
using assms
proof (induct length $p$ arbitrary: $p$ $v$)
  case 0
  hence $s = v$ by auto
  thus ?case using 0(1) general-source-val
    by (metis awalk-cost-Nil length-0-conv zero-ereal-def)
next
  case (Suc $n$)
  then obtain $p' \ e$ where $p' :: p @ [e]$
    by (cases $p$ rule: rev-cases) auto
  then obtain $u$ where ewu: awalk s $p' \ u \land \text{awalk} \ u \ [e] \ v$
    using awalk-append-iff Suc(3) by simp
  then have $du$: $\text{dist} \ u \leq \text{ereal} \ (\text{awalk-cost} \ c \ p')$
    using Suc $p' \ e$ by simp
  from ewu have $ust$: $u = \text{tail} \ G \ e$ and $vta$: $v = \text{head} \ G \ e$
    by auto
  then have $\text{dist} \ v \leq \text{dist} \ u + c \ e$
    using ewu du ust trian[where $e=e$] by force
  with $du$ have $\text{dist} \ v \leq \text{ereal} \ (\text{awalk-cost} \ c \ p') + c \ e$
    by (metis add-right-mono order-trans)
  thus $\text{dist} \ v \leq \text{awalk-cost} \ c \ p$
    using awalk-cost-append $p' \ e$ by simp
qed

lemma (in fin-digraph) witness-path:
assumes $\mu \ c \ s \ v = \text{ereal } r$
shows $\exists \ p. \ \text{apath } s \ p \ v \land \mu \ c \ s \ v = \text{awalk-cost } c \ p$
proof –
  have $sv: s \rightarrow^* v$
    using shortest-path-inf[of $s \ v \ c$] assms by fastforce
  { 
    fix $p$ assume awalk $s \ p \ v$
    then have no-neg-cyc:
      $\neg (\exists w \ q. \ \text{awalk } w \ q \ w \land w \in \text{set (awalk-verts } s \ p) \land \text{awalk-cost } c \ q < 0)$
      using neg-cycle-imp-inf-$\mu$ assms by force
  }
  thus $\neg \text{thesis}$ using no-neg-cyc-reach-imp-path[$OF \ sv$] by presburger
qed

lemma (in basic-sp) dist-le-$\mu$:
  fixes $v :: 'a$
  assumes $v \in \text{verts } G$
  shows $\text{dist } v \leq \mu \ c \ s \ v$
proof (rule ccontr)
  assume $nt: \neg \text{thesis}$
  show $\text{False}$
proof (cases $\mu \ c \ s \ v$)
  show $\forall r. \ \mu \ c \ s \ v = \text{ereal } r \implies \text{False}$
    proof –
      fix $r$ assume $r$-asm: $\mu \ c \ s \ v = \text{ereal } r$
      hence $sv: s \rightarrow^* v$
      using shortest-path-inf[where $u=s$ and $v=v$ and $f=c$] by auto
      obtain $p$ where
        awalk $s \ p \ v$
        $\mu \ c \ s \ v = \text{awalk-cost } c \ p$
        using witness-path[$OF \ r$-asm] unfolding apath-def by force
      thus $\text{False}$ using $nt$ dist-le-cost by simp
    qed
  next
  show $\mu \ c \ s \ v = \infty \implies \text{False}$ using $nt$ by simp
  next
  show $\mu \ c \ s \ v = - \infty \implies \text{False}$ using dist-le-cost
    proof –
      assume asm: $\mu \ c \ s \ v = - \infty$
      let $?C = (\lambda x. \ \text{ereal (awalk-cost } c \ x)) \cdot \{p. \ \text{awalk } s \ p \ v\}$
      have $\exists \ x. \ x < \text{dist } v$
        using $nt$ unfolding $\mu$-def not-le INF-less-iff by simp
      then obtain $p$ where
        awalk $s \ p \ v$
        awalk-cost $c \ p < \text{dist } v$
        by force
      thus $\text{False}$ using dist-le-cost by force
    qed
  qed
  qed
lemma (in basic-just-sp) dist-ge-µ:
  fixes v :: 'a
  assumes v ∈ verts G
  assumes num v ≠ ∞
  assumes dist v ≠ −∞
  assumes µ c s s = ereal 0
  assumes dist s = 0
  assumes ∃ u. u ∈ verts G ⇒ u ≠ s ⇒ num u ≠ ∞ ⇒ num u ≠ enat 0
  shows dist v ≥ µ c s v
proof –
  obtain n where enat n = num v using assms(2) by force
  thus ?thesis using assms
proof (induct n arbitrary: v)
next
  case (Suc n)
  thus ?case by (cases v=s, auto)
  case False
  obtain e where e-assms:
    e ∈ arcs G
    v = head G e
    dist v = dist (tail G e) + ereal (c e)
    num v = num (tail G e) + enat 1
  using just[OF Suc(3) False Suc(4)] by blast
then have nsinf: num (tail G e) ≠ ∞
  by (metis Suc(2) enat.simps(3) enat-1 plus-enat-simps(2))
then have ns: enat n = num (tail G e)
  using e-assms(4) Suc(2) by force
have ds: dist (tail G e) = µ c s (tail G e)
  using Suc(1)[OF ns tail-in-verts[OF e-assms(1)] nsinf]
  Suc(5−8) e-assms(3) dist-le-µ[OF tail-in-verts[OF e-assms(1)]]
  by simp
have dmuc: dist v ≥ µ c s (tail G e) + ereal (c e)
  using e-assms(3) ds by auto
thus ?thesis
proof (cases dist v = ∞)
  case False
  have arc-to-ends G e = (tail G e, v)
    unfolding arc-to-ends-def
    by (simp add: e-assms(2))
  obtain r where µr: µ c s (tail G e) = ereal r
    using e-assms(3) Suc(5) ds False
    by (cases µ c s (tail G e), auto)
  obtain p where
    awalk s p (tail G e) and
$$\mu s \cdot \mu c s (\text{tail } G e) = \text{ereal} (\text{awalk-cost } c p)$$

using witness-path(OF $\mu r$) unfolding apath-def
by blast

then have pe: awalk $s (p @ [e]) v$
using e-assms(1,2) by (auto simp: awalk-simps)

hence muc: $\mu c s v \leq \mu c s (\text{tail } G e) + \text{ereal} (c e)$
using $\mu$s min-cost-le-walk-cost(OF pe) by simp

thus dist $v \geq \mu c s v$ using dmuc by simp

qed simp

qed (simp add: Suc(6,7))

qed

lemma (in shortest-path-pos-cost) tail-value-check:

fixes $u :: \alpha$

assumes $s \in \text{verts } G$

shows $\mu c s s = \text{ereal } 0$

proof

have $\ast$: awalk $s [] s$
using assms unfolding awalk-def by simp

hence $\mu c s s \leq \text{ereal } 0$
using min-cost-le-walk-cost(OF $\ast$) by simp

moreover

have $(\forall p. \text{awalk } s p s \Rightarrow \text{ereal}(\text{awalk-cost } c p) \geq \text{ereal } 0)$
using pos-cost pos-cost-pos-awalk-cost by auto

hence $\mu c s s \geq \text{ereal } 0$

unfolding $\mu$-def by (blast intro: INF-greatest)

ultimately

show ?thesis by simp

qed

lemma (in shortest-path-pos-cost) num-not0:

fixes $v :: \alpha$

assumes $v \in \text{verts } G$

assumes $v \neq s$

assumes num $v \neq \infty$

shows num $v \neq \text{enat } 0$

proof

obtain $ku$ where num $v = ku + \text{enat } 1$
using assms just by blast

thus ?thesis by (induct $ku$) auto

qed

lemma (in shortest-path-pos-cost) dist-ne-ninf:

fixes $v :: \alpha$

assumes $v \in \text{verts } G$

shows dist $v \neq -\infty$

proof (cases num $v = \infty$)
case False
obtain $n$ where enat $n = \text{num } v$
using False by force
thus \textit{thesis using assms False}

\textbf{proof (induct n arbitrary: v)}

\textbf{case 0} thus \textit{?case}

using num-not0 tail-val by (cases v=s, auto)

\textbf{next}

\textbf{case (Suc n)}

thus \textit{?case}

\textbf{proof (cases v=s)}

\textbf{case True}

thus \textit{thesis using tail-val by simp}

\textbf{next}

\textbf{case False}

obtain \(e\) where \(e\text{-assms:}\)

\(e \in \text{arcs } G\)

\(\text{dist } v = \text{dist } (\text{tail } G e) + \text{ereal } (c e)\)

\(\text{num } v = \text{num } (\text{tail } G e) + \text{enat } 1\)

\textbf{using just[OF Suc(3) False Suc(4)] by blast}

then have \(\text{nsinf: num } (\text{tail } G e) \neq \infty\)

by (metis Suc(2) enat.simps(3) enat-1 plus-enat-simps(2))

then have \(\text{ns: enat } n = \text{num } (\text{tail } G e)\)

\textbf{using e-assms(3) Suc(2) by force}

\textbf{have dist (tail G e) \neq -\infty}

by (rule Suc(1) [OF ns tail-in-verts[OF e-assms(1) nsinf]])

\textbf{thus \textit{thesis using e-assms(2) by simp}}

\textbf{qed}

\textbf{qed}

\textbf{next}

\textbf{case True}

\textbf{thus \textit{thesis using no-path[OF assms] by simp}}

\textbf{qed}

\textbf{theorem (in shortest-path-pos-cost)} \textit{correct-shortest-path:}

\textbf{fixes } v :: 'a

\textbf{assumes } v \in \text{verts } G

\textbf{shows } \text{dist } v = \mu c s v

\textbf{using no-path[OF assms(1)] dist-le-\mu[OF assms(1)] dist-ge-\mu[OF assms(1)] - dist-ne-ninf[OF assms(1)]}

\textbf{tail-value-check[OF s-in-G tail-val num-not0]}

\textbf{by fastforce}

\textbf{corollary (in shortest-path-pos-cost-pred)} \textit{correct-shortest-path-pred:}

\textbf{fixes } v :: 'a

\textbf{assumes } v \in \text{verts } G

\textbf{shows } \text{dist } v = \mu c s v

\textbf{using correct-shortest-path assms by simp}

\textbf{end}

\textbf{theory ShortestPathNeg}
imports ShortestPath

begin

2 Shortest Path (with general edge costs)

locale shortest-paths-locale-step1 =
  fixes G :: ('a, 'b) pre-digraph (structure)
  fixes s :: 'a
  fixes c :: 'b ⇒ real
  fixes num :: 'a ⇒ nat
  fixes parent-edge :: 'a ⇒ 'b option
  fixes dist :: 'a ⇒ ereal
assumes graphG: fin-digraph G
assumes s-assms: s ∈ verts G
dist s ≠ ∞
parent-edge s = None
num s = 0
assumes parent-num-assms:
  ⋀v. [v ∈ verts G; v ≠ s; dist v ≠ ∞] →
  (∃e ∈ arcs G. parent-edge v = Some e ∧
  head G e = v ∧ dist (tail G e) ≠ ∞ ∧
  num v = num (tail G e) + 1)
assumes noPedge: ⋀e. e ∈ arcs G →
dist (tail G e) ≠ ∞ → dist (head G e) ≠ ∞

sublocale shortest-paths-locale-step1 ⊆ fin-digraph G
using graphG by auto

definition (in shortest-paths-locale-step1) enum :: 'a ⇒ enat where
  enum v = (if (dist v = ∞ ∨ dist v = -∞) then ∞ else num v)

locale shortest-paths-locale-step2 =
  shortest-paths-locale-step1 +
  basic-just-sp G dist c s enum +
assumes source-val: (∃v ∈ verts G. enum v ≠ ∞) → dist s = 0
assumes no-edge-Vm-Vf:
  ⋀e. e ∈ arcs G → dist (tail G e) = -∞ → ∀ r. dist (head G e) ≠ ereal r

function (in shortest-paths-locale-step1) pwalk :: 'a ⇒ 'b list
where
  pwalk v =
  (if (v = s ∨ dist v = ∞ ∨ v ∉ verts G)
  then []
  else pwalk (tail G (the (parent-edge v))) @ [the (parent-edge v)]
) by auto
termination (in shortest-paths-locale-step1)
using parent-num-assms
by (relation measure num, auto, fastforce)

lemma (in shortest-paths-locale-step1) pwalk-simps:
v = s ∨ dist v = ∞ ∨ v ∉ verts G ⇒ pwalk v = []
v ≠ s ⇒ dist v ≠ ∞ ⇒ v ∈ verts G ⇒
pwalk v = pwalk (tail G (the (parent-edge v))) @ [the (parent-edge v)]
by auto

definition (in shortest-paths-locale-step1) pwalk-verts :: 'a set where
pwalk-verts v = {u. u ∈ set (awalk-verts s (pwalk v))}

locale shortest-paths-locale-step3 =
shortest-paths-locale-step2 +
fixes C :: ('a × ('b awalk)) set
assumes C-se: C ⊆ {(u, p), dist u ≠ ∞ ∧ awalk u p u ∧ awalk-cost c p < 0}
assumes int-neg-cyc:
∀v, v ∈ verts G ⇒ dist v = −∞ ⇒
(fst ' C) ∩ pwalk-verts v ≠ {}

locale shortest-paths-locale-step2-pred =
shortest-paths-locale-step1 +
fixes pred :: 'a ⇒ 'b option
assumes bj: basic-just-sp-pred G dist c s enum pred
assumes source-val: (∃v ∈ verts G. enum v ≠ ∞) ⇒ dist s = 0
assumes no-edge-Vm-Vf:
∀e. e ∈ arcs G ⇒ dist (tail G e) = −∞ ⇒ ∀r. dist (head G e) ≠ ereal r

lemma (in shortest-paths-locale-step1) num-s-is-min:
assumes v ∈ verts G
assumes v ≠ s
assumes dist v ≠ ∞
shows num v > 0
using parent-num-assms[OF assms] by fastforce

lemma (in shortest-paths-locale-step1) path-from-root-Vr-ex:
fixes v :: 'a
assumes v ∈ verts G
assumes v ≠ s
assumes dist v ≠ ∞
shows ∃e. s →* tail G e ∧
e ∈ arcs G ∧ head G e = v ∧ dist (tail G e) ≠ ∞ ∧
parent-edge v = Some e ∧ num v = num (tail G e) + 1
using assms
proof (induct num v − 1 arbitrary : v)
case 0
  obtain e where ee:
  \( e \in \text{arcs } G \) head \( G \) e = v dist (tail \( G \) e) \( \neq \infty \)
  parent-edge \( v = \text{Some } e \) num v = num (tail \( G \) e) + 1
  using parent-num-assms[\( \text{OF } 0(2-4) \)] by fast
  have tail \( G \) e = s
  \( \text{using num-s-is-min[\( \text{OF } \text{tail-in-verts[OF ee(1)]} - \text{ee(3)} \)]} \)
  \( \text{ex(5) } 0(1) \) by auto
  then show ?case using ee by auto
next
case (Suc \( n' \))
  obtain e where ee:
  \( e \in \text{arcs } G \) head \( G \) e = v dist (tail \( G \) e) \( \neq \infty \)
  parent-edge \( v = \text{Some } e \) num v = num (tail \( G \) e) + 1
  using parent-num-assms[\( \text{OF } \text{Suc}(3-5) \)] by fast
  then have ss: tail \( G \) e \( \neq s \)
  \( \text{using num-s-is-min tail-in-verts} \)
  \( \text{Suc}(2) \) s-assms(4) by force
  have nst: \( n' = \text{num } (\text{tail } G \ e) - 1 \)
  \( \text{using ee(5) Suc(2) by presburger} \)
  obtain e' where reach: \( s \to^* \text{tail } G \ e' \) and
  \( e'; \ e' \in \text{arcs } G \) head \( G \) e' = tail \( G \) e dist (tail \( G \) e') \( \neq \infty \)
  using Suc(1)[\( \text{OF } nst \text{tail-in-verts[OF ee(1)] ss ee(3)} \)] by blast
  then have s \( \to^* \text{tail } G \ e \)
  \( \text{by (metis arc-implies-awalk reachable-awalk reachable-trans)} \)
  then show ?case using e' ee by auto
qed

lemma (in shortest-paths-locale-step1) path-from-root-Vr:
  fixes v :: 'a
  assumes v \( \in \) verts \( G \)
  assumes dist v \( \neq \infty \)
  shows s \( \to^* v \)
proof(cases v = s)
case True thus \( \text{thesis using } \text{assms by simp} \)
next
case False
  obtain e where s \( \to^* \text{tail } G \ e \in \text{arcs } G \) head \( G \) e = v
  using path-from-root-Vr-ex[\( \text{OF } \text{assms(1) False } \text{assms(2)} \)] by blast
  then have s \( \to^* \text{tail } G \ e \text{ tail } G \ e \to v \)
  \( \text{by (auto intro: in-arcs-imp-in-arcs-ends)} \)
  then show \( \text{thesis by (rule reachable-adj-trans)} \)
qed

lemma (in shortest-paths-locale-step1) \( \mu \)-V-less-inf:
  fixes v :: 'a
  assumes v \( \in \) verts \( G \)
  assumes dist v \( \neq \infty \)
  shows \( \mu \ c \ s \ v \neq \infty \)
using assms path-from-root-Vr μ-reach-conv by force

lemma (in shortest-paths-locale-step2) enum-not0:
  assumes v ∈ verts G
  assumes v ≠ s
  assumes enum v ≠ ∞
  shows enum v ≠ enat 0
  using parent-num-assms[OF assms(1,2)] assms unfolding enum-def by auto

lemma (in shortest-paths-locale-step2) dist-Vf-µ:
  fixes v :: 'a
  assumes vG: v ∈ verts G
  assumes ∃ r. dist v = ereal r
  shows dist v = µ c s v
proof (cases v = s)
  case True
  thus ?thesis using assms pwalk.simps[where v=v] awalk-Nil-iff by presburger
next
  case False
from assms show ?thesis
proof (induct rule: pwalk.induct)
  fix v
  let ?e = the (parent-edge v)
  let ?u = tail G ?e
  assume ewu: ¬ (v = s ∨ dist v = ∞ ∨ v ∉ verts G) ⟹
  ?u ∈ verts G ⟹ dist ?u ≠ ∞ ⟹
  awalk s (pwalk ?u) ?u
assume $vG: v \in \text{verts } G$
assume $dv: \text{dist } v \neq \infty$
thus $\text{awalk } s (\text{pwalk } v) v$

proof (cases $v = s \lor \text{dist } v = \infty \lor v \notin \text{verts } G$)
case True
   thus "$\ddagger$thesis$
   using \text{pwalk-simps } vG \ dv$
   awalk-Nil-iff by fastforce

next
case False
obtain $e \ where ee:
   e \in \text{arcs } G$
parent-edge $v = \text{Some } e$
head $G \ e = v$
dist (tail $G \ e) \neq \infty$
using parent-num-assms False by blast
hence $\text{awalk } s (\text{pwalk } ?u) ?u$
using $\text{ewu[OF False]} \ \text{tail-in-verts by simp}$
hence $\text{awalk } s (\text{pwalk } (\text{tail } G \ e) \ @ \ [e]) v$
using $ee(1-3) vG$
by (auto simp: awalk-simps simp del: pwalk.simps)
also have $\text{pwalk } (\text{tail } G \ e) \ @ \ [e] = \text{pwalk } v$
using $\text{False } ee(2) \ \text{unfolding pwalk.simps[where } v=v] \ by \ auto$
finally show "$\ddagger$thesis$.$
qed

lemma (in shortest-paths-locale-step3) $\mu$-ninf:
   fixes $v :: \ 'a$
   assumes $v \in \text{verts } G$
   assumes $\text{dist } v = -\infty$
   shows $\mu \ c \ s \ v = -\infty$
proof
   have $\text{awalk } s (\text{pwalk } v) v$
   using $\text{pwalk-awalk } \text{assms by force}$
moreover
obtain $w \ where \ ww: w \in \text{fst } C \cap \text{pwalk-verts } v$
   using int-neg-cyc[of $\text{assms}$] by blast
then obtain $q \ where$
   awalk $w \ q \ w$
   awalk-cost $c \ q < 0$
   using $C-se$ by auto
moreover
   have $w \in \text{set } (\text{awalk-verts } s (\text{pwalk } v))$
   using $\text{ww } \text{unfolding pwalk-verts-def by fast}$
ultimately
   show "$\ddagger$thesis$ using \text{neg-cycle-imp-inf-} \mu \ by \ force$
qed
lemma (in shortest-paths-locale-step3) correct-shortest-path:
  fixes v :: 'a
  assumes v ∈ verts G
  shows dist v = μ c s v
proof (cases dist v)
show ∀ r. dist v = ereal r → dist v = μ c s v
  using dist-Vf-[OF assms] by simp
next
show dist v = ∞ → dist v = μ c s v
  using μ-V-less-inf [OF assms] by simp
next
show dist v = −∞ → dist v = μ c s v
  using μ-ninf [OF assms] by simp
qed

end

References