An Axiomatic Characterization of the Single-Source Shortest Path Problem

By Christine Rizkallah

September 19, 2015

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 \( \mu : V \to \mathbb{R} \cup \{\infty\} \) 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 \( c : E \to \mathbb{R} \). 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
Complex
../Graph-Theory/Graph-Theory
~~~/src/HOL/Library/Extended-Nat
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:
\( \forall e. \, e \in \text{arcs } G \quad \Rightarrow \quad \text{dist } (\text{head } G \, e) \leq \text{dist } (\text{tail } G \, e) + c \, e \)

locale basic-just-sp = 
basic-sp +
fixes num :: 'a ⇒ enat
assumes just:
\( \forall v. \, [v \in \text{verts } G; \, v \neq s; \, \text{num } v \neq \infty] \quad \Rightarrow \quad \exists e \in \text{arcs } G. \, v = \text{head } G \, e \land \quad \text{dist } v = \text{dist } (\text{tail } G \, e) + c \, e \land \quad \text{num } v = \text{num } (\text{tail } G \, e) + (\text{enat } 1) \)

locale shortest-path-pos-cost = 
basic-just-sp +
assumes s-in-G: s ∈ verts G
assumes tail-val: dist s = 0
assumes no-path: \( \forall v. \, v \in \text{verts } G \quad \Rightarrow \quad \text{dist } v = \infty \quad \iff \quad \text{num } v = \infty \)
assumes pos-cost: \( \forall e. \, e \in \text{arcs } G \quad \Rightarrow \quad 0 \leq c \, e \)

locale basic-just-sp-pred = 
basic-sp +
fixes num :: 'a ⇒ enat
fixes pred :: 'a ⇒ 'b option
assumes just:
\( \forall v. \, [v \in \text{verts } G; \, v \neq s; \, \text{num } v \neq \infty] \quad \Rightarrow \quad \exists e \in \text{arcs } G. \quad e = \text{the } (\text{pred } v) \land \quad v = \text{head } G \, e \land \quad \text{dist } v = \text{dist } (\text{tail } G \, e) + c \, e \land \quad \text{num } v = \text{num } (\text{tail } G \, e) + (\text{enat } 1) \)

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 =
basic-just-sp-pred

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

sublocale shortest-path-pos-cost-pred ⊆ 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 ≠ []
shows p = [hd p]
by (metis assms distinct
.simps(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 list
assumes awalk s p v
shows dist v ≤ 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 = p’ @ [e]
     by (cases p rule: rev-cases) auto
  then obtain u where ewu: awalk s p’ u ∧ awalk u [e] v
     using awalk-append-iff Suc(3) by simp
  then have du: dist u ≤ereal (awalk-cost c p’)
     using Suc p’ e by simp
  from ewu have ust: u = tail G e and eta: v = head G e
     by auto
  then have dist v ≤ dist u + c e
     using ewu du ust \( \text{trian} \ [\text{where} \ c = e] \) by force
  with du have dist v ≤ereal (awalk-cost c p’) + c e
     by (metis add-right-mono order-trans)
  thus dist v ≤ 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 \( \text{shortest-path-inf[of } s v c \] \) assms by fastforce
  { 
    fix \( p \) assume \( \text{awalk } s p v \)
    then have no-neg-cyc:
      \( \neg (\exists w q. \text{awalk } w q w \land w \in \text{set } (\text{awalk-verts } s p) \land \text{awalk-cost } c q < 0) \)
      using \( \text{neg-cycle-imp-inf-\mu assms by force} \)
  }
thus \( \neg \text{thesis} \) using \( \text{no-neg-cyc-reach-imp-path[OF } sv \] \) by presburger
qed

lemma (in basic-sp) \( \text{dist-le-\mu:} \)
  fixes \( v :: 'a \)
  assumes \( v \in \text{verts } G \)
  shows \( \text{dist } v \leq \mu c s v \)
proof (rule ccontr)
  assume \( nt: \neg \neg \text{thesis} \)
  show \( \text{False} \)
    proof (cases \( \mu c s v \))
      next
      show \( \mu c s v = \infty \Rightarrow \text{False} \) using \( nt \) by simp
    next
      show \( \mu c s v = -\infty \Rightarrow \text{False} \) using dist-le-cost
    proof
      assume asm: \( \mu c s v = -\infty \)
      let \( ?C = (\lambda x. \text{ereal } (\text{awalk-cost } c x)) \) \( \cdot \) \( \{ p. \text{awalk } s p v \} \)
      have \( \exists x \in ?C. x < \text{dist } v \)
        using \( nt \) unfolding \( \mu \text{-def not-le INF-less-iff} \) by simp
      then obtain \( p \) where
        \( \text{awalk } s p v \)
        \( \text{awalk-cost } c p < \text{dist } v \)
        by force
      thus \( \neg \text{thesis} \) 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)
case 0 thus ?case by (cases v=s, auto)
next
case (Suc n)
  thus ?case
proof (cases v=s)
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
\[ \mu \cdot \mu \cdot s \cdot (\text{tail } G \cdot e) = \epsilon \cdot \text{ereal} \cdot (\text{awalk-cost } c \cdot p) \]

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

then have \( p e : \text{awalk } s \cdot (p @ [e]) \cdot v \)
  using e-assms(1,2) by (auto simp: awalk-simps)

hence \( \mu u : \mu \cdot c \cdot s \cdot v \leq \mu \cdot c \cdot s \cdot (\text{tail } G \cdot e) + \epsilon \cdot \text{ereal } (c \cdot e) \)
  using \( \mu u \cdot \text{min-cost-le-walk-cost}[OF \ p e] \) by simp

thus \( \text{dist } v \geq \mu \cdot c \cdot s \cdot v \) using dmuc by simp
qed simp

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

qed

lemma (in shortest-path-pos-cost) tail-value-check:
  fixes \( u : 'a \)
  assumes \( s \in \text{verts } G \)
  shows \( \mu \cdot c \cdot s \cdot s = \epsilon \cdot \text{ereal } 0 \)
proof –
  have \( * : \text{awalk } s \cdot [] \cdot s \) using assms unfolding awalk-def by simp
  hence \( \mu \cdot c \cdot s \cdot s \leq \epsilon \cdot \text{ereal } 0 \) using min-cost-le-walk-cost[OF *] by simp

moreover
  have \( (\forall p. \text{awalk } s \cdot p \cdot s \implies \epsilon \cdot \text{ereal } (\text{awalk-cost } c \cdot p) \geq \epsilon \cdot \text{ereal } 0) \)
  using pos-cost pos-cost-pos-awalk-cost by auto

hence \( \mu \cdot c \cdot s \cdot s \geq \epsilon \cdot \text{ereal } 0 \)
  unfolding \( \mu \cdot \text{def} \) by (blast intro: INF-greatest)

ultimately
  show \( \text{thesis} \) by simp
qed

lemma (in shortest-path-pos-cost) num-not0:
  fixes \( v : 'a \)
  assumes \( v \in \text{verts } G \)
  assumes \( v \neq s \)
  assumes \( \text{num } v \neq \infty \)
  shows \( \text{num } v \neq \text{enat } 0 \)
proof –
  obtain \( k u \) where \( \text{num } v = k u + \text{enat } 1 \)
  using False by force

  using assms just by blast

  thus \( \text{thesis} \) by (induct \( k u \)) auto

qed

lemma (in shortest-path-pos-cost) dist-ne-ninf:
  fixes \( v : 'a \)
  assumes \( v \in \text{verts } G \)
  shows \( \text{dist } v \neq -\infty \)
proof (cases \( \text{num } v = \infty \))
  case False
  obtain \( n \) where \( \text{enat } n = \text{num } v \)
  using False by force

qed
thus \(?thesis\) using \(assms\) False

proof (induct \(n\) arbitrary: \(v\))

case 0 thus \(?case\)
using num-not0 tail-val by (cases \(v\)=s, auto)

next
case (Suc \(n\))
thus \(?case\)
proof (cases \(v\)=s)

case True
thus \(?thesis\) using tail-val by simp

next
case False
obtain \(e\) where \(e\)-assms:
\(e\) \(\in\) arcs \(G\)
\(dist\) \(v\) = \(dist\) (tail \(G\) \(e\)) + \(\text{ereal}\) \((c\) \(e\))
\(num\) \(v\) = \(num\) (tail \(G\) \(e\)) + \(enat\) \(1\)
using just\([OF\ Suc(3)\ False\ Suc(4)]\) by blast
then have nsinf\:\(\text{num}\) (tail \(G\) \(e\)) \(\neq\) \(\infty\)
by (metis Suc\([2]\) enat.simps\([3]\) enat-1 plus-enat-simps\([2]\))
then have ns\:\(\text{enat}\) \(\text{num}\) \(\text{num}\) (tail \(G\) \(e\)) + \(\text{enat}\) \(1\)
using e-assms\([3]\) Suc\([2]\) by force
have dist (tail \(G\) \(e\)) \(\neq\) \(\infty\)
by (rule Suc\([1]\) \([OF\ ns\ tail-in-verts[OF\ e-assms(1)]\ nsinf]\))
thus \(?thesis\) using e-assms\([2]\) by simp

qed

next
case True
thus \(?thesis\) using \(\text{no-path}[OF\ assms]\) by simp

qed

end
2 Shortest Path (with general edge costs)

locale shortest-paths-locale-step1 =
fixes $G :: (\text{a}', \text{b})$ pre-digraph (structure)
fixes $s :: \text{a}'$
fixes $c :: \text{b} \Rightarrow \text{real}$
fixes $\text{num} :: \text{a}' \Rightarrow \text{nat}$
fixes $\text{parent-edge} :: \text{a} \Rightarrow \text{b} \Rightarrow \text{option}$
fixes $\text{dist} :: \text{a} \Rightarrow \text{ereal}$
assumes $\text{graphG} :: \text{fin-digraph G}$
assumes $s\text{-assms}$:
$s \in \text{verts G}$
dist $s \neq \infty$
parent-edge $s = \text{None}$
num $s = 0$
assumes $\text{parent-num-assms}$:
$\forall v. [v \in \text{verts G}; v \neq s; \text{dist} v \neq \infty] \implies$
$(\exists e \in \text{arcs G}. \text{parent-edge} v = \text{Some } e \land$
head $G e = v \land \text{dist} (\text{tail} G e) \neq \infty \land$
num $v = \text{num} (\text{tail} G e) + 1)$
assumes $\text{noPedge}$:
$\forall e. e \in \text{arcs G} \implies$
dist (tail $G e$) $\neq \infty \implies \text{dist} (\text{head} G e) \neq \infty$

sublocale shortest-paths-locale-step1 $\subseteq \text{fin-digraph G}$
using $\text{graphG}$ by auto

definition (in shortest-paths-locale-step1) $\text{enum} :: \text{a}' \Rightarrow \text{enat}$ where
$\text{enum} v = (\text{if } (\text{dist} v = \infty \lor \text{dist} v = -\infty) \text{ then } \infty \text{ else num } v)$

locale shortest-paths-locale-step2 =
shortest-paths-locale-step1 +
basic-just-sp $G$ dist $c$ $s$ enum +
assumes $\text{source-val}$:
$(\exists v \in \text{verts G}. \text{enum} v \neq \infty) \implies \text{dist} s = 0$
assumes $\text{no-edge-Vm-Vf}$:
$\forall e. e \in \text{arcs G} \implies \text{dist} (\text{tail} G e) = -\infty \implies \forall r. \text{dist} (\text{head} G e) \neq \text{ereal } r$

function (in shortest-paths-locale-step1) $\text{pwalk} :: \text{a} \Rightarrow \text{b} \Rightarrow \text{list}$ where
$pwalk v =$
$(\text{if } (v = s \lor \text{dist} v = \infty \lor v \notin \text{verts G})$
then $[]$
else $\text{pwalk} (\text{tail} G (\text{the} (\text{parent-edge} v))) \circ \text{[the} (\text{parent-edge} v)]$
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 × (′a awalk)) set
assumes C-sc:
  C ⊆ { (u, p). dist u ≠ ∞ ∧ awalk u p ∧ 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

9
using \texttt{assms}

\textbf{proof} (\texttt{induct num v \textasciitilde 1 arbitrary : v})

\texttt{case 0}

\texttt{obtain e where ee:
  e \in arcs G head G e = v dist (tail G e) \neq \infty
  parent-edge v = \texttt{Some e num v = num (tail G e) + 1
  using parent-num-assms[OF 0(2\textasciitilde4)] by fast
}

\texttt{have tail G e = s
  using num-s-is-min[OF tail-in-verts [OF ee(1)] - ee(3)]
  ee(5) 0(1) by auto
}

\texttt{then show ?case using ee by auto}

\texttt{next}

\texttt{case (Suc n')}

\texttt{obtain e where ee:
  e \in arcs G head G e = v dist (tail G e) \neq \infty
  parent-edge v = \texttt{Some e num v = num (tail G e) + 1
  using parent-num-assms[OF Suc(3\textasciitilde5)] by fast
}

\texttt{then have ss: tail G e \neq s
  using num-s-is-min tail-in-verts
  Suc(2) s-assms(4) by force
}

\texttt{have nst: n' = num (tail G e) \textasciicircum 1
  using ee(5) Suc(2) by presburger
}

\texttt{obtain e' where reach: s \rightarrow^* tail G e' and
  e': e' \in arcs G head G e' = tail G e dist (tail G e') \neq \infty
  using Suc(1)[OF nst tail-in-verts[OF ee(1)] ss ee(3)] by blast
}

\texttt{then have s \rightarrow^* tail G e
  by (metis arc-implies-awalk reachable-awalk reachable-trans)
}

\texttt{then show ?case using e' ee by auto}

\texttt{qed}

\textbf{lemma (in shortest-paths-locale-step1) path-from-root-Vr:}

\texttt{fixes v :: 'a
  assumes v \in verts G
  assumes dist v \neq \infty
  shows s \rightarrow^* v
}

\textbf{proof(cases v = s)}

\texttt{case True thus \texttt{thesis using assms by simp
}

\texttt{next}

\texttt{case False}

\texttt{obtain e where s \rightarrow^* tail G e e \in arcs G head G e = v
  using path-from-root-Vr-ex[OF assms(1) False assms(2)] by blast
}

\texttt{then have s \rightarrow^* tail G e tail G e \rightarrow v
  by (auto intro: in-arcs-imp-in-arcs-ends
}

\texttt{then show \texttt{thesis by (rule reachable-adj-trans)
}

\texttt{qed}

\textbf{lemma (in shortest-paths-locale-step1) \mu-V-less-inf:}

\texttt{fixes v :: 'a
  assumes v \in verts G}
assumes \( \text{dist } v \neq \infty \)
shows \( \mu \ c \ s \ v \neq \infty \)
using \( \text{assms } \text{path-from-root-Vr } \mu\text{-reach-conv } \text{by force} \)

lemma (in shortest-paths-locale-step2) \text{enum-not0}:
assumes \( v \in \text{verts } G \)
assumes \( v \neq s \)
assumes \( \text{enum } v \neq \infty \)
shows \( \text{enum } v \neq \text{enat } 0 \)
using \( \text{parent-num-assms} \)[OF \( \text{assms} \)] \( \text{assms } \text{unfolding } \text{enum-def } \text{by } \text{auto} \)

lemma (in shortest-paths-locale-step2) \text{dist-Vf-}\( \mu \):
fixes \( v :: 'a \)
assumes \( vG : v \in \text{verts } G \)
assumes \( \exists r \cdot \text{dist } v = \text{ereal } r \)
shows \( \text{dist } v = \mu \ c \ s \ v \)
proof −
have \( \text{ds: } \text{dist } s = 0 \)
using \( \text{assms source-val } \text{unfolding } \text{enum-def } \text{by force} \)
have \( \text{ews:awalk s }[] \ s \)
using \( \text{s-assms(1)} \) unfolding \( \text{awalk-def } \text{by simp} \)
have \( \text{mu: } \mu \ c \ s \ s = \text{ereal } 0 \)
using \( \text{min-cost-le-walk-cost}[OF \text{ews, where } c=c] \)
awalk-cost-nil \( \text{ds dist-le-}\mu[\text{OF } \text{s-assms}(1)] \) zero-ereal-def
by simp
thus \( \text{thesis} \)
using \( \text{ds assms dist-le-}\mu[\text{OF } vG] \)
dist-ge-\( \mu[\text{OF } vG - - \text{mu ds enum-not0}] \)
unfolding \( \text{enum-def } \text{by fastforce} \)

qed

lemma (in shortest-paths-locale-step1) \text{pwalk-awalk}:
fixes \( v :: 'a \)
assumes \( v \in \text{verts } G \)
assumes \( \text{dist } v \neq \infty \)
shows \( \text{awalk s (pwalk } v \text{)} \ v \)
proof (cases \( v=s \))
case True
thus \( \text{thesis} \)
using \( \text{assms } \text{pwalk.simps}[\text{where } v=v] \)
awalk-nil-iff by presburger
next
case False
from \( \text{assms } \text{show } \text{thesis} \)
proof (induct rule: \( \text{pwalk.induct} \))
fix \( v \)
let \( ?e = \text{the (parent-edge } v \text{)} \)
let \( ?u = \text{tail } G \ ?e \)
assume \( \text{euw: } \neg (v = s \lor \text{dist } v = \infty \lor v \notin \text{verts } G) \implies \)
Lemma (in shortest-paths-locale-step3) µ-ninf:
\[ \mu c s v = -\infty \]

Proof:

1. \( \exists u \in \text{verts } G \implies \text{dist } \exists u \neq \infty \implies \text{awalk } (\text{pwalk } \exists u) \exists u \)

2. Assume \( v \in \text{verts } G \)
3. Assume \( d_v \text{ dist } v \neq \infty \)
4. Thus \( \text{awalk } (\text{pwalk } v) \)

Proof (cases \( v = s \lor \text{dist } v = \infty \lor v \notin \text{verts } G \))

Case True

   Thus \( ?\text{thesis} \)

   Using \( \text{pwalk.} \text{simps } v G d_v \)

   awalk-Nil-iff by fastforce

Next

Case False

Obtain \( e \) where \( ee \):

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

parent-edge \( v = \text{Some } e \)

head \( G e = v \)

\( \text{dist } (\text{tail } G e) \neq \infty \)

Using parent-num-assms False by blast

Hence \( \text{awalk } (\text{pwalk } \exists u) \exists u \)

Using \( \text{exv[OF False] tail-in-verts by simp } \)

Hence \( \text{awalk } (\text{pwalk } (\text{tail } G e) \circ [e]) \)

Using \( ee(1-3) v G \)

By (auto simp: awalk-simps simp del: pwalk.\text{simps})

Also have \( \text{pwalk } (\text{tail } G e) \circ [e] = \text{pwalk } v \)

Using False \( ee(2) \) unfolding pwalk.\text{simps}[where \( v=v \)] by auto

Finally show ?\text{thesis} .

qed

qed

Lemma (in shortest-paths-locale-step3) µ-ninf:

\[ \mu c s v = -\infty \]

Proof:

1. Obtain \( \exists u \) where \( uu \):

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

parent-edge \( v = \text{Some } e \)

head \( G e = v \)

\( \text{dist } (\text{tail } G e) \neq \infty \)

Using parent-num-assms False by blast

Hence \( \text{awalk } (\text{pwalk } \exists u) \exists u \)

Using \( \text{exv[OF False] tail-in-verts by simp } \)

Hence \( \text{awalk } (\text{pwalk } (\text{tail } G e) \circ [e]) \)

Using \( ee(1-3) v G \)

By (auto simp: awalk-simps simp del: pwalk.\text{simps})

Also have \( \text{pwalk } (\text{tail } G e) \circ [e] = \text{pwalk } v \)

Using False \( ee(2) \) unfolding pwalk.\text{simps}[where \( v=v \)] by auto

Finally show ?\text{thesis} .

qed

qed

lemma (in shortest-paths-locale-step3) µ-ninf:

\[ \mu c s v = -\infty \]

Proof:

1. Obtain \( \exists u \) where \( uu \):

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

parent-edge \( v = \text{Some } e \)

head \( G e = v \)

\( \text{dist } (\text{tail } G e) \neq \infty \)

Using parent-num-assms False by blast

Hence \( \text{awalk } (\text{pwalk } \exists u) \exists u \)

Using \( \text{exv[OF False] tail-in-verts by simp } \)

Hence \( \text{awalk } (\text{pwalk } (\text{tail } G e) \circ [e]) \)

Using \( ee(1-3) v G \)

By (auto simp: awalk-simps simp del: pwalk.\text{simps})

Also have \( \text{pwalk } (\text{tail } G e) \circ [e] = \text{pwalk } v \)

Using False \( ee(2) \) unfolding pwalk.\text{simps}[where \( v=v \)] by auto

Finally show ?\text{thesis} .

qed

qed
show ?thesis using neg-cycle-imp-inf-\( \mu \) by force

qed

lemma (in shortest-paths-locale-step3) correct-shortest-path:
  fixes v :: 'a
  assumes v \in verts G
  shows dist v = \( \mu \) c s v
proof (cases dist v)
  show \( \forall r. \ dist v = ereal r \implies dist v = \mu \) c s v
    using dist-Vf-\( \mu \)[OF assms] by simp
  next
  show dist v = \( \infty \) \implies dist v = \( \mu \) c s v
    using \( \mu \)-V-less-inf[OF assms]
    dist-le-\( \mu \)[OF assms] by simp
  next
  show dist v = \( -\infty \) \implies dist v = \( \mu \) c s v
    using \( \mu \)-ninf[OF assms] by simp
qed

end

References