

# An Axiomatic Characterization of the Single-Source Shortest Path Problem

By Christine Rizkallah

May 26, 2024

## 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

<b>1</b>	<b>Shortest Path (with non-negative edge costs)</b>	<b>2</b>
<b>2</b>	<b>Shortest Path (with general edge costs)</b>	<b>8</b>
	<i>theory ShortestPath</i>	
	<b>imports</b>	
	<i>Graph-Theory.Graph-Theory</i>	
	<b>begin</b>	

# 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  $\Rightarrow$  ereal
  fixes c :: 'b  $\Rightarrow$  real
  fixes s :: 'a
  assumes general-source-val:  $dist\ s \leq 0$ 
  assumes trian:
     $\bigwedge e. e \in arcs\ G \implies$ 
       $dist\ (head\ G\ e) \leq dist\ (tail\ G\ e) + c\ e$ 

```

```

locale basic-just-sp =
  basic-sp +
  fixes num :: 'a  $\Rightarrow$  enat
  assumes just:
     $\bigwedge v. \llbracket v \in verts\ G; v \neq s; num\ v \neq \infty \rrbracket \implies$ 
       $\exists e \in arcs\ G. v = head\ G\ e \wedge$ 
         $dist\ v = dist\ (tail\ G\ e) + c\ e \wedge$ 
         $num\ v = num\ (tail\ G\ e) + (enat\ 1)$ 

```

```

locale shortest-path-pos-cost =
  basic-just-sp +
  assumes s-in-G:  $s \in verts\ G$ 
  assumes tail-val:  $dist\ s = 0$ 
  assumes no-path:  $\bigwedge v. v \in verts\ G \implies dist\ v = \infty \longleftrightarrow num\ v = \infty$ 
  assumes pos-cost:  $\bigwedge e. e \in arcs\ G \implies 0 \leq c\ e$ 

```

```

locale basic-just-sp-pred =
  basic-sp +
  fixes num :: 'a  $\Rightarrow$  enat
  fixes pred :: 'a  $\Rightarrow$  'b option
  assumes just:
     $\bigwedge v. \llbracket v \in verts\ G; v \neq s; num\ v \neq \infty \rrbracket \implies$ 
       $\exists e \in arcs\ G.$ 
         $e = the\ (pred\ v) \wedge$ 
         $v = head\ G\ e \wedge$ 
         $dist\ v = dist\ (tail\ G\ e) + c\ e \wedge$ 
         $num\ v = num\ (tail\ G\ e) + (enat\ 1)$ 

```

```

sublocale basic-just-sp-pred  $\subseteq$  basic-just-sp
using basic-just-sp-pred-axioms
unfolding basic-just-sp-pred-def
  basic-just-sp-pred-axioms-def
by unfold-locale (blast)

```

```

locale shortest-path-pos-cost-pred =

```

*basic-just-sp-pred* +  
**assumes** *s-in-G*:  $s \in \text{verts } G$   
**assumes** *tail-val*:  $\text{dist } s = 0$   
**assumes** *no-path*:  $\bigwedge v. v \in \text{verts } G \implies \text{dist } v = \infty \longleftrightarrow \text{num } v = \infty$   
**assumes** *pos-cost*:  $\bigwedge e. e \in \text{arcs } G \implies 0 \leq c \ e$

**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.simps(2) list.sel(1) neq-Nil-conv last-ConsR last-in-set*)

**lemma** (**in** *basic-sp*) *dist-le-cost*:  
**fixes**  $v :: 'a$   
**fixes**  $p :: 'b \text{ list}$   
**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**  $p'e: p = p' @ [e]$   
  **by** (*cases p rule: rev-cases*) *auto*  
  **then obtain**  $u$  **where**  $ewu: \text{awalk } s \ p' \ u \wedge \text{awalk } u \ [e] \ v$   
  **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  $\exists p. \text{apath } s p v \wedge \mu c s v = \text{awalk-cost } c p$ 
proof –
  have  $sv: s \rightarrow^* v$ 
  using shortest-path-inf[of s v c] assms by fastforce
  {
    fix  $p$  assume  $\text{awalk } s p v$ 
    then have no-neg-cyc:
       $\neg (\exists w q. \text{awalk } w q w \wedge w \in \text{set } (\text{awalk-verts } s p) \wedge \text{awalk-cost } c q < 0)$ 
    using neg-cycle-imp-inf- $\mu$  assms by force
  }
  thus ?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  $nt: \neg ?thesis$ 
  show False
  proof (cases  $\mu c s v$ )
    show  $\bigwedge r. \mu c s v = \text{ereal } r \implies \text{False}$ 
    proof –
      fix  $r$  assume  $r\text{-asm}: \mu c s v = \text{ereal } r$ 
      hence  $sv: s \rightarrow^* v$ 
      using shortest-path-inf[where u=s and v=v and f=c] by auto
      obtain  $p$  where
         $\text{awalk } s p v$ 
         $\mu c s v = \text{awalk-cost } c p$ 
      using witness-path[OF r-asm] unfolding apath-def by force
      thus False using nt dist-le-cost by simp
    qed
  next
  show  $\mu c s v = \infty \implies \text{False using } nt \text{ by simp}$ 
  next
  show  $\mu c s v = -\infty \implies \text{False using } \text{dist-le-cost}$ 
  proof –
    assume  $asm: \mu c s v = -\infty$ 
    let  $?C = (\lambda x. \text{ereal } (\text{awalk-cost } c x)) \text{ ' } \{p. \text{awalk } s p v\}$ 
    have  $\exists x \in ?C. x < \text{dist } v$ 
    using nt unfolding  $\mu$ -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 False using dist-le-cost by force
  qed
qed

```

qed

lemma (in basic-just-sp) dist-ge- $\mu$ :

fixes  $v :: 'a$   
assumes  $v \in \text{verts } G$   
assumes  $\text{num } v \neq \infty$   
assumes  $\text{dist } v \neq -\infty$   
assumes  $\mu \text{ c s s} = \text{ereal } 0$   
assumes  $\text{dist } s = 0$   
assumes  $\bigwedge u. u \in \text{verts } G \implies u \neq s \implies$   
 $\text{num } u \neq \infty \implies \text{num } u \neq \text{enat } 0$   
shows  $\text{dist } v \geq \mu \text{ c s } v$

proof –

obtain  $n$  where  $\text{enat } n = \text{num } v$  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*:

$e \in \text{arcs } G$

$v = \text{head } G \ e$

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

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

using *just[OF Suc(3) False Suc(4)]* by blast

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

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

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

using *e-assms(4) Suc(2)* by force

have *ds*:  $\text{dist } (\text{tail } G \ e) = \mu \text{ c s } (\text{tail } G \ e)$

using *Suc(1)[OF ns tail-in-verts[OF e-assms(1)] nsinf]*

*Suc(5–8) e-assms(3) dist-le- $\mu$ [OF tail-in-verts[OF e-assms(1)]]*

by *simp*

have *dmuc*:  $\text{dist } v \geq \mu \text{ c s } (\text{tail } G \ e) + \text{ereal } (c \ e)$

using *e-assms(3) ds* by auto

thus ?thesis

proof (cases  $\text{dist } v = \infty$ )

case False

have *arc-to-ends*  $G \ e = (\text{tail } G \ e, v)$

unfolding *arc-to-ends-def*

by (*simp add: e-assms(2)*)

obtain  $r$  where  $\mu \text{ c s } (\text{tail } G \ e) = \text{ereal } r$

using *e-assms(3) Suc(5) ds False*

by (cases  $\mu \text{ c s } (\text{tail } G \ e)$ , auto)

obtain  $p$  where

*awalk*  $s \ p$  ( $\text{tail } G \ e$ ) and

```

     $\mu s: \mu c s (tail\ G\ e) = ereal\ (awalk\text{-}cost\ c\ p)$ 
    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: $\mu c s v \leq \mu c s (tail\ G\ e) + ereal\ (c\ e)$ 
    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
qed

```

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

```

  fixes u :: 'a
  assumes s  $\in$  verts G
  shows  $\mu c s s = ereal\ 0$ 
proof -
  have *: awalk s [] s using assms unfolding awalk-def by simp
  hence  $\mu c s s \leq ereal\ 0$  using min-cost-le-walk-cost[OF *] by simp
  moreover
  have ( $\bigwedge p. awalk\ s\ p\ s \implies ereal(awalk\text{-}cost\ c\ p) \geq ereal\ 0$ )
    using pos-cost pos-cost-pos-awalk-cost by auto
  hence  $\mu c s s \geq ereal\ 0$ 
    unfolding  $\mu$ -def by (blast intro: INF-greatest)
  ultimately
  show ?thesis by simp
qed

```

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

```

  fixes v :: 'a
  assumes v  $\in$  verts G
  assumes v  $\neq$  s
  assumes num v  $\neq$   $\infty$ 
  shows num v  $\neq$  enat 0
proof -
  obtain ku where num v = ku + 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 :: 'a
  assumes v  $\in$  verts G
  shows dist v  $\neq$   $-\infty$ 
proof (cases num v =  $\infty$ )
case False
  obtain n where enat n = num v
    using False by force

```

```

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 ∈ arcs G
      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(3) Suc(2) by force
    have dist (tail G e) ≠ - ∞
      by (rule Suc(1) [OF ns tail-in-verts[OF e-assms(1)] nsinf])
    thus ?thesis using e-assms(2) by simp
  qed
  qed
next
case True
  thus ?thesis using no-path[OF assms] by simp
qed

theorem (in shortest-path-pos-cost) correct-shortest-path:
  fixes v :: 'a
  assumes v ∈ verts G
  shows dist v =  $\mu$  c s v
  using no-path[OF assms(1)] dist-le- $\mu$ [OF assms(1)]
    dist-ge- $\mu$ [OF assms(1)] - dist-ne-ninf[OF assms(1)]
    tail-value-check[OF s-in-G] tail-val num-not0
  by fastforce

corollary (in shortest-path-pos-cost-pred) correct-shortest-path-pred:
  fixes v :: 'a
  assumes v ∈ verts G
  shows dist v =  $\mu$  c s v
  using correct-shortest-path assms by simp

end
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**  $c :: 'b \Rightarrow \text{real}$

**fixes**  $\text{num} :: 'a \Rightarrow \text{nat}$

**fixes**  $\text{parent-edge} :: 'a \Rightarrow 'b$  *option*

**fixes**  $\text{dist} :: 'a \Rightarrow \text{ereal}$

**assumes**  $\text{graphG}$ : *fin-digraph*  $G$

**assumes**  $s\text{-assms}$ :

$s \in \text{verts } G$

$\text{dist } s \neq \infty$

$\text{parent-edge } s = \text{None}$

$\text{num } s = 0$

**assumes**  $\text{parent-num-assms}$ :

$\bigwedge v. \llbracket v \in \text{verts } G; v \neq s; \text{dist } v \neq \infty \rrbracket \Longrightarrow$

$(\exists e \in \text{arcs } G. \text{parent-edge } v = \text{Some } e \wedge$

$\text{head } G e = v \wedge \text{dist } (\text{tail } G e) \neq \infty \wedge$

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

**assumes**  $\text{noPedge}$ :  $\bigwedge e. e \in \text{arcs } G \Longrightarrow$

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

**sublocale** *shortest-paths-locale-step1*  $\subseteq$  *fin-digraph*  $G$

**using**  $\text{graphG}$  **by** *auto*

**definition** (in *shortest-paths-locale-step1*)  $\text{enum} :: 'a \Rightarrow \text{enat}$  **where**

$\text{enum } v = (\text{if } (\text{dist } v = \infty \vee \text{dist } v = -\infty) \text{ then } \infty \text{ else } \text{num } v)$

**locale** *shortest-paths-locale-step2* =

*shortest-paths-locale-step1* +

*basic-just-sp*  $G$   $\text{dist } c$   $s$   $\text{enum}$  +

**assumes**  $\text{source-val}$ :  $(\exists v \in \text{verts } G. \text{enum } v \neq \infty) \Longrightarrow \text{dist } s = 0$

**assumes**  $\text{no-edge-Vm-Vf}$ :

$\bigwedge e. e \in \text{arcs } G \Longrightarrow \text{dist } (\text{tail } G e) = -\infty \Longrightarrow \forall r. \text{dist } (\text{head } G e) \neq \text{ereal } r$

**function** (in *shortest-paths-locale-step1*)  $\text{pwalk} :: 'a \Rightarrow 'b$  *list*

**where**

$\text{pwalk } v =$

$(\text{if } (v = s \vee \text{dist } v = \infty \vee v \notin \text{verts } G)$

$\text{then } []$

$\text{else } \text{pwalk } (\text{tail } G (\text{the } (\text{parent-edge } v))) \ @ \ [\text{the } (\text{parent-edge } v)])$

$)$

**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*:  
 $v = s \vee \text{dist } v = \infty \vee v \notin \text{verts } G \implies \text{pwalk } v = []$   
 $v \neq s \implies \text{dist } v \neq \infty \implies v \in \text{verts } G \implies$   
 $\text{pwalk } v = \text{pwalk } (\text{tail } G \text{ (the (parent-edge } v))) \text{ @ [the (parent-edge } v)]$   
 by *auto*

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

**locale** *shortest-paths-locale-step3* =  
*shortest-paths-locale-step2* +  
**fixes**  $C :: ('a \times ('b \text{ awalk})) \text{ set}$   
**assumes** *C-se*:  
 $C \subseteq \{(u, p). \text{dist } u \neq \infty \wedge \text{awalk } u \text{ } p \text{ } u \wedge \text{awalk-cost } c \text{ } p < 0\}$   
**assumes** *int-neg-cyc*:  
 $\bigwedge v. v \in \text{verts } G \implies \text{dist } v = -\infty \implies$   
 $(\text{fst } ' C) \cap \text{pwalk-verts } v \neq \{\}$

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

**lemma** (in *shortest-paths-locale-step1*) *num-s-is-min*:  
**assumes**  $v \in \text{verts } G$   
**assumes**  $v \neq s$   
**assumes**  $\text{dist } v \neq \infty$   
**shows**  $\text{num } v > 0$   
 using *parent-num-assms*[*OF assms*] **by** *fastforce*

**lemma** (in *shortest-paths-locale-step1*) *path-from-root-Vr-ex*:  
**fixes**  $v :: 'a$   
**assumes**  $v \in \text{verts } G$   
**assumes**  $v \neq s$   
**assumes**  $\text{dist } v \neq \infty$   
**shows**  $\exists e. s \rightarrow^* \text{tail } G \text{ } e \wedge$   
 $e \in \text{arcs } G \wedge \text{head } G \text{ } e = v \wedge \text{dist } (\text{tail } G \text{ } e) \neq \infty \wedge$   
 $\text{parent-edge } v = \text{Some } e \wedge \text{num } v = \text{num } (\text{tail } G \text{ } e) + 1$   
**using** *assms*  
**proof**(*induct num v - 1 arbitrary : v*)

```

case 0
  obtain e where ee:
    e ∈ arcs G head G e = v dist (tail G e) ≠ ∞
    parent-edge v = Some e num v = num (tail G e) + 1
    using parent-num-assms[OF 0(2-4)] by fast
  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
  then show ?case using ee by auto
next
case (Suc n')
  obtain e where ee:
    e ∈ arcs G head G e = v dist (tail G e) ≠ ∞
    parent-edge v = Some e num v = num (tail G e) + 1
    using parent-num-assms[OF Suc(3-5)] by fast
  then have ss: tail G e ≠ s
    using num-s-is-min tail-in-verts
    Suc(2) s-assms(4) by force
  have nst: n' = num (tail G e) - 1
    using ee(5) Suc(2) by presburger
  obtain e' where reach: s →* tail G e' and
    e': e' ∈ arcs G head G e' = tail G e dist (tail G e') ≠ ∞
    using Suc(1)[OF nst tail-in-verts[OF ee(1)] ss ee(3)] by blast
  then have s →* tail G e
    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 ∈ verts G
  assumes dist v ≠ ∞
  shows s →* v
proof(cases v = s)
case True thus ?thesis using assms by simp
next
case False
  obtain e where s →* tail G e e ∈ arcs G head G e = v
    using path-from-root-Vr-ex[OF assms(1) False assms(2)] by blast
  then have s →* tail G e tail G e → v
    by (auto intro: in-arcs-imp-in-arcs-ends)
  then show ?thesis by (rule reachable-adj-trans)
qed

```

```

lemma (in shortest-paths-locale-step1) μ-V-less-inf:
  fixes v :: 'a
  assumes v ∈ verts G
  assumes dist v ≠ ∞
  shows μ c s v ≠ ∞

```

```

using assms path-from-root-Vr μ-reach-conv by force

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

lemma (in shortest-paths-locale-step1) pwalk-awalk:
  fixes  $v :: 'a$ 
  assumes  $v \in \text{verts } G$ 
  assumes  $\text{dist } v \neq \infty$ 
  shows  $\text{awalk } s \ (\text{pwalk } v) \ v$ 
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  $?e = \text{the } (\text{parent-edge } v)$ 
    let  $?u = \text{tail } G \ ?e$ 
    assume  $ewu: \neg (v = s \vee \text{dist } v = \infty \vee v \notin \text{verts } G) \implies$ 
       $?u \in \text{verts } G \implies \text{dist } ?u \neq \infty \implies$ 
       $\text{awalk } s \ (\text{pwalk } ?u) \ ?u$ 

```

```

assume  $vG: v \in \text{verts } G$ 
assume  $dv: \text{dist } v \neq \infty$ 
thus  $\text{awalk } s \text{ (pwalk } v) v$ 
proof ( $\text{cases } v = s \vee \text{dist } v = \infty \vee v \notin \text{verts } G$ )
case True
  thus ?thesis
    using  $\text{pwalk.simps } vG dv$ 
     $\text{awalk-Nil-iff}$  by fastforce
next
case False
  obtain  $e$  where  $ee:$ 
     $e \in \text{arcs } G$ 
     $\text{parent-edge } v = \text{Some } e$ 
     $\text{head } G e = v$ 
     $\text{dist } (\text{tail } G e) \neq \infty$ 
    using  $\text{parent-num-assms } False$  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 False ee(2) unfolding pwalk.simps[where v=v] by auto
  finally show ?thesis .
qed
qed
qed

lemma (in shortest-paths-locale-step3)  $\mu\text{-ninf}$ :
  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  $\text{pwalk-awalk assms}$  by force
moreover
  obtain  $w$  where  $ww: w \in \text{fst } 'C \cap \text{pwalk-verts } v$ 
    using  $\text{int-neg-cyc}[OF assms]$  by blast
  then obtain  $q$  where
     $\text{awalk } w q w$ 
     $\text{awalk-cost } c q < 0$ 
    using  $C\text{-se}$  by auto
moreover
  have  $w \in \text{set } (\text{awalk-verts } s \text{ (pwalk } v))$ 
    using  $ww$  unfolding  $\text{pwalk-verts-def}$  by fast
ultimately
  show ?thesis using  $\text{neg-cycle-imp-inf-}\mu$  by force
qed

```

```

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

end

```

## References

- [1] E. Alkassar, S. Böhme, K. Mehlhorn, and C. Rizkallah. A framework for the verification of certifying computations. *Journal of Automated Reasoning*, 2013. To Appear.