An Axiomatic Characterization of the Single-Source Shortest Path Problem

By Christine Rizkallah

August 16, 2018

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) 4

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 
analyses 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 
analyses 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 ⟨proof⟩ 
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: \( \forall e. e \in \text{arcs } G \implies 0 \leq e \)

sublocale shortest-path-pos-cost-pred \( \subseteq \) shortest-path-pos-cost

⟨proof⟩

lemma tail-value-helper:
assumes hd \( p = \) last \( p \)
assumes distinct \( p \)
assumes \( p \neq [] \)
shows \( p = [\text{hd } p] \)
⟨proof⟩

lemma (in basic-sp) dist-le-cost:
fixes \( v :: 'a \)
fixes \( p :: 'b \) list
assumes awalk \( s p v \)
shows \( \text{dist } v \leq \text{awalk-cost } c p \)
⟨proof⟩

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⟩

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⟩

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 \( \mu c s s = \text{ereal } 0 \)
assumes \( \text{dist } s = 0 \)
assumes \( \forall u. u \in \text{verts } G \implies u \neq s \implies \) \( \text{num } u \neq \infty \implies \text{num } u \neq \text{enat } 0 \)
shows \( \text{dist } v \geq \mu c s v \)
⟨proof⟩

lemma (in shortest-path-pos-cost) tail-value-check:
fixes \( a :: 'a \)
assumes \( s \in \text{verts } G \)
shows \( \mu c s s = \text{ereal } 0 \)
⟨proof⟩
lemma (in shortest-path-pos-cost) num-not0:
  fixes v :: 'a
  assumes v ∈ verts G
  assumes v ≠ s
  assumes num v ≠ ∞
  shows num v ≠ enat 0
⟨proof⟩

lemma (in shortest-path-pos-cost) dist-ne-ninf:
  fixes v :: 'a
  assumes v ∈ verts G
  shows dist v ≠ −∞
⟨proof⟩

theorem (in shortest-path-pos-cost) correct-shortest-path:
  fixes v :: 'a
  assumes v ∈ verts G
  shows dist v = µ c s v
⟨proof⟩

corollary (in shortest-path-pos-cost-pred) correct-shortest-path-pred:
  fixes v :: 'a
  assumes v ∈ verts G
  shows dist v = µ c s v
⟨proof⟩

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 ⇒ 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:
\[ \forall v. \ [v \in \text{verts } G; v \neq s; \ dist v \neq \infty] \implies \]
\[ (\exists e \in \text{arcs } G. \ \text{parent-edge } v = \text{Some } e \wedge \]
\[ \text{head } G e = v \wedge \dist (\text{tail } G e) \neq \infty \wedge \]
\[ \text{num } v = \text{num } (\text{tail } G e) + 1) \]
assumes noPedge: \[ \forall e. \ e \in \text{arcs } G \implies \]
\[ \dist (\text{tail } G e) \neq \infty \implies \dist (\text{head } G e) \neq \infty \]

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

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

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

function (in shortest-paths-locale-step1) pwalk :: 'a \Rightarrow 'b list
where
\[ \text{pwalk } v = \]
\[ (\text{if } (v = s \lor \dist v = \infty \lor v \notin \text{verts } G) \]
\[ \text{then } [] \]
\[ \text{else } \text{pwalk } (\text{tail } G (\text{the } \text{parent-edge } v))) \# \text{[the } \text{parent-edge } v] \]
\[ ) \]

termination (in shortest-paths-locale-step1)

lemma (in shortest-paths-locale-step1) pwalk-simps:
\[ v = s \lor \dist v = \infty \lor v \notin \text{verts } G \implies \text{pwalk } v = [] \]
\[ v \neq s \implies \dist v \neq \infty \implies v \in \text{verts } G \implies \]
\[ \text{pwalk } v = \text{pwalk } (\text{tail } G (\text{the } \text{parent-edge } v))) \# \text{[the } \text{parent-edge } v] \]

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

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

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

**shortest-paths-locale-step1** +

**fixes** \(\text{pred} :: 'a \Rightarrow 'b\) **option**

**assumes** \(b\text{j}::\text{basic-just-sp-pred} G\) dist \(c\) \(s\) 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:

\[\forall e.\ e \in \text{arcs} G \implies \text{dist} (\text{tail} G\ e) = -\infty \implies \forall r.\ \text{dist} (\text{head} G\ e) \neq \text{ereal} \ r\]

**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** \(\text{num} \ v > 0\)

\being proof \endproof

**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** \(\exists e.\ s \rightarrow^* \text{tail} G\ e \land\ e \in \text{arcs} G \land \text{head} G\ e = v \land \text{dist} (\text{tail} G\ e) \neq \infty \land \text{parent-edge} v = \text{Some} \ e \land \text{num} v = \text{num} (\text{tail} G\ e) + 1\)

\being proof \endproof

**lemma** (in shortest-paths-locale-step1) path-from-root-Vr:

**fixes** \(v::'a\)

**assumes** \(v \in \text{verts} G\)

**assumes** \(\text{dist} v \neq \infty\)

**shows** \(s \rightarrow^* v\)

\being proof \endproof

**lemma** (in shortest-paths-locale-step1) \(\mu\)-V-less-inf:

**fixes** \(v::'a\)

**assumes** \(v \in \text{verts} G\)

**assumes** \(\text{dist} v \neq \infty\)

**shows** \(\mu c\ s\ v \neq \infty\)

\being proof \endproof

**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** \(\text{enum} v \neq \text{enat} \ 0\)

\being proof \endproof
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⟩
lemma (in shortest-paths-locale-step1) pwalk-awalk:
  fixes v :: 'a
  assumes v ∈ verts G
  assumes dist v ≠ ∞
  shows awalk s (pwalk v) v
⟨proof⟩
lemma (in shortest-paths-locale-step3) μ-ninf:
  fixes v :: 'a
  assumes v ∈ verts G
  assumes dist v = −∞
  shows μ c s v = −∞
⟨proof⟩
lemma (in shortest-paths-locale-step3) correct-shortest-path:
  fixes v :: 'a
  assumes v ∈ verts G
  shows dist v = μ c s v
⟨proof⟩
end

References