Suggesting Edits to Explain Failing Traces

Giles Reger

University of Manchester, Manchester, UK

September 25, 2015

Outline

Motivation

- For property $(ab^*c)^*$ which trace is *more wrong*?

  1. $a.b.c.a.b.b.c.a.b.c.a.b.b.c.a.b$
  2. $a.a.a.a.a.c.c.c.c.c.a.a.a.a.a.c.c.c.c.c$

- Both traces violate the property, but that's not very informative
- We want a better measure for violation

- How could the first trace be *fixed*?
  - Add a c to the end
  - Remove the last a.b
  - Replace last b by c
- How many *edits* required to fix the second trace?

Reminder: Edit Distance

The edit (Levenshtein) distance between traces τ_1 and τ_2 is distance(τ_1, τ_2), defined as

$$\begin{aligned}
\text{distance}(\tau_1, \epsilon) &= |\tau_1| \\
\text{distance}(\epsilon, \tau_2) &= |\tau_2| \\
\text{distance}(a\tau_1, b\tau_2) &= \min \begin{cases} \text{distance}(\tau_1, b\tau_2) + 1 & \\ \text{distance}(a\tau_1, \tau_2) + 1 & \\ \text{distance}(\tau_1, \tau_2) + 1 & \text{if } a \neq b \\ \text{distance}(\tau_1, \tau_2) & \text{if } a = b \end{cases}
\end{aligned}$$

The edit distance between a trace τ and an automaton φ is the smallest distance between τ and a trace in the language of φ

$$\text{distance}(\tau, \varphi) = \min(\{\text{distance}(\tau, \tau') \mid \tau' \in \mathcal{L}(\varphi)\})$$

Edit Distance as a Verdict

- Typically in RV we have a specification φ and trace τ and ask

$$\varphi \stackrel{?}{\in} \mathcal{L}(\varphi)$$

  The answer can be 'yes' or 'no'
- Replacing with

$$\text{distance}(\tau, \varphi) =?$$

  can give more information, in certain settings
- If $\text{distance}(\tau, \varphi) = 0$ then $\varphi \in \mathcal{L}(\varphi)$

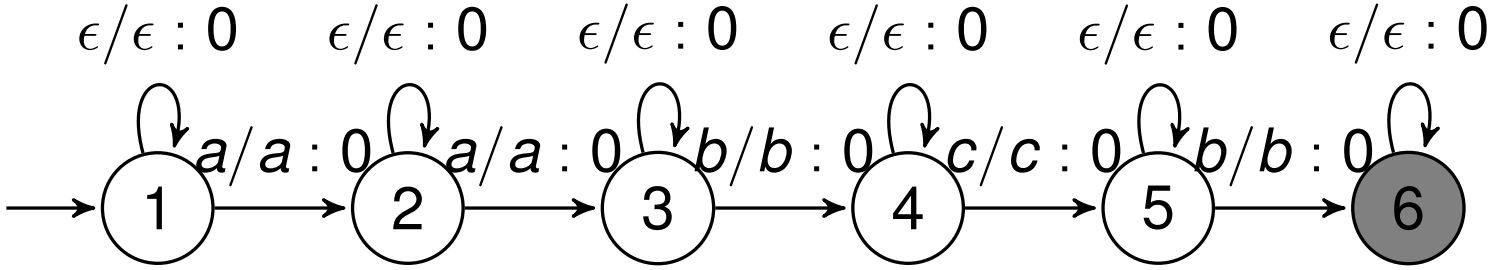
- Applications include
  - Specification learning (fitness function, imperfect traces)
  - Violation explanations
  - Repair

Edits as Explanations

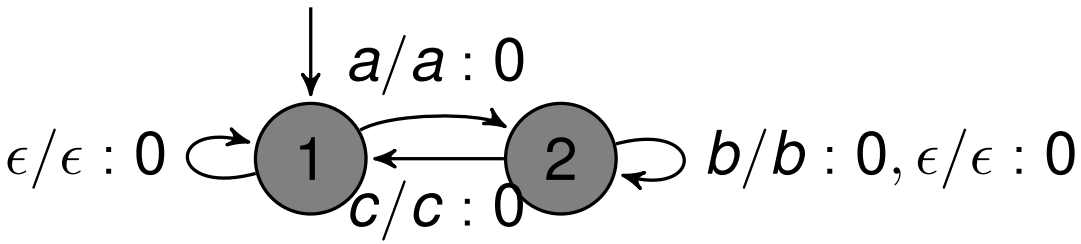
- When computing the edit distance you get the edits required for that distance for free
- These edit operations can be used to explain *why* the trace violates the property
- The shortest edit distance may not be the best explanation
- And there may be many sets of edits that give the shortest distance
- Heuristics are required

Computing Edits using Transducers

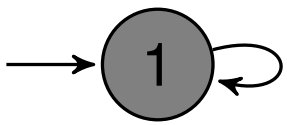
- Idea to use *weighted transducers* by Allauzen and Mohri
- The trace $a.a.b.c.b$ would be



- The property $(ab^*c)^*$ would be

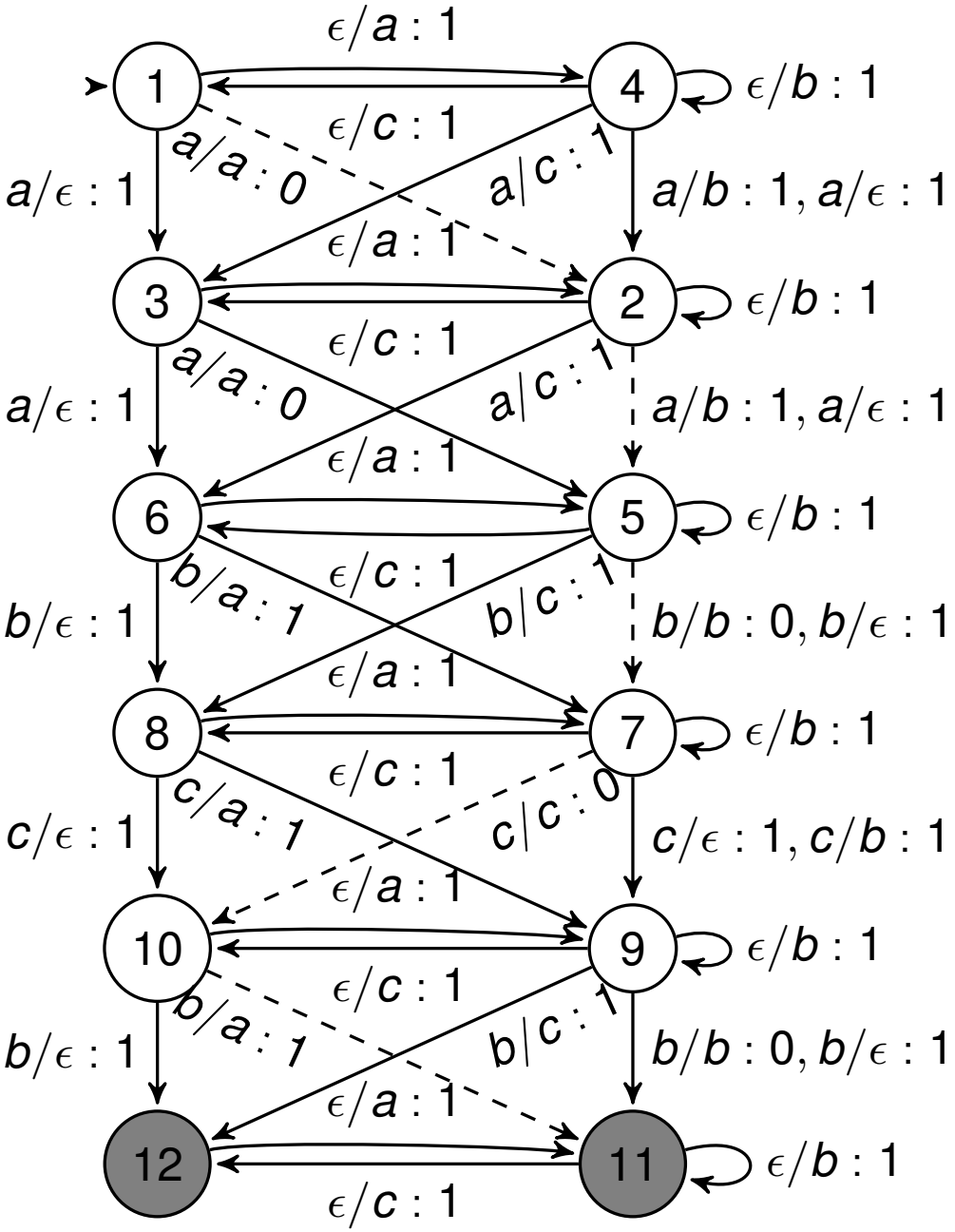


- And the edits would be captured as



$$a/a : 0, b/b : 0, c/c : 0,$$
$$a/\epsilon : 1, b/\epsilon : 1, c/\epsilon : 1, \epsilon/a : 1, \epsilon/b : 1, \epsilon/c : 1$$
$$a/b : 1, a/c : 1, b/a : 1, b/c : 1, c/a : 1, c/b : 1$$

Computing Edits using Transducers



- Then compute their 3-way composition
- $T \circ E \circ P$
- A path to a final state is an edit
- The shortest such path is the edit distance

Does it make sense to edit a trace?

open.close.open.open.close.open.close.open.open.close
A_1 A_2 A_3 B_1 B_2 A_1 A_2 A_3 C_1 C_3

- Editing position A_1 effects two points in the trace
- We should not edit one A_1 without editing the other

- Label the trace and make edits consistent with labels

- What is a minimal edit path now?
  1. Add *close* after A_3
  2. Add *close* before B_1 and before C_1
  3. Remove *open* at B_1 and C_1
- Want 1 to be smaller than 2 or 3 as edits fewer labels
- Also may prefer certain operations i.e. 2 preferred to 3

Labelled Edits

- Labelled Event is a pair of an event and a label
- Can update composition operation to preserve labels
- An *edit record* is $((\langle a_1, l \rangle, a_2, w)$
- An *edit path* is a finite sequence of edit records starting (ending) in an initial (accepting) state of $T \circ E \circ P$
- A sensible *edit path*
  1. Applies edits consistently wrt labels
  2. Minimises the number of labels effected
- The *cost* of an edit path τ is given as $\text{cost}(\tau, \{\})$ defined as $\text{cost}(\epsilon, S) = 0$ and $\text{cost}(((\langle a_1, l_1 \rangle, a_2, w).\tau, S) =$

$$\text{cost}(\tau, S + (a_1/a_2, l_1)) + \begin{cases} w & \text{if } (a_1/a_2, l_1) \notin S \\ 0 & \text{if } (a_1/a_2, l_1) \in S \end{cases}$$

Labelled Edits

- Labelled Event is a pair of an event and a label
- Can update composition operation to preserve labels
- An *edit record* is $((\langle a_1, l \rangle, a_2, w)$
- An *edit path* is a finite sequence of edit records starting (ending) in an initial (accepting) state of $T \circ E \circ P$
- A sensible *edit path*
  1. Applies edits consistently wrt labels
  2. Minimises the number of labels effected
- The *cost* of an edit path τ is given as $\text{cost}(\tau, \{\})$ defined as $\text{cost}(\epsilon, S) = 0$ and $\text{cost}(((\langle a_1, l_1 \rangle, a_2, w).\tau, S) =$

$$\text{cost}(\tau, S + (a_1/a_2, l_1)) + \begin{cases} w & \text{if } (a_1/a_2, l_1) \notin S \\ 0 & \text{if } (a_1/a_2, l_1) \in S \end{cases}$$

Heuristic Search

- Use *heuristic search* to find sensible edit paths
  1. Follow 0-weighted path modulo consistency
  2. Choose a (short) path to closest state with 0-weight transition
  3. If in final state return path, otherwise goto 1

- Based on the assumption that deviations will be infrequent and short

- Obvious exponential branching nature
- However, tamed by necessity to preserve consistency

- Can perform search with limit on edit distance
- Found that searching with $max = 0, 1, 2, ...$ helpful

Conclusion

- Had idea that edit distance would be useful for RV
- Obviously wasn't the first - see related work

- Lots of directions to explore
  - More expressive automata?
  - For LTL... how to relate explanations to property
  - Extend to quantified properties... extra dimension
    - Consider numeric constraints on quantifiers?
  - Refine notion of relating edits to trace source
  - Better heuristics
  - Better implementation, naive implementation in Scala
  - Method for detecting *multiple errors*?