Testen ohne Testorakel

Oliver Ebsen

Universität Hamburg

28. Juni 2017

Inhaltsverzeichnis

Nicht testbar

- Programm zur Findung des Ergebnisses geschrieben.
- Überprüfen der Ergebnisse zu teuer.
- Überprüfen der Ergebnisse macht diese obsolet (Prognoseprogramme).
- Übersichtsartikel: [Weyuker, 1982]
- Formalisierung der Orakelannahme und maschinelles Lernen: [Kanewala and Bieman, 2013]
- Metamorphisches Testen: [Tsong Y Chen and Yiu, 1998]

Testsystem

Definition

Unter einem Testsystem verstehen wir ein Tupel $(P, S, T, O, corr, corr_t)$ wobei

- P eine Menge von Programmen ist,
- S eine Menge von Programmspezifikationen ist,
- T eine Menge von Tests ist,
- O eine Menge von Orakeln o ist, und jedes o eine Teilmenge von $T \times P$ ist, nämlich genau der Art, dass $o(t, p)$ für ein $p \in P$ und ein $t \in T$ bestimmt, ob p t besteht.
- Außerdem ist $corr \subseteq P \times S$ derart, dass $corr(p, s)$ mit $p \in P$ und $s \in S$ bedeutet, dass p die Spezifikation s erfüllt.
- Schlussendlich ist $corr_t \subseteq T \times P \times S$ derart, dass $corr_t(t, p, s)$ bedeutet, dass ein Programm $p \in P$ die Spezifikation $s \in S$ erfüllt, die im Test $t \in T$ getestet werden.

Die Orakelannahme

Definition (korrektes Orakel, vollständiges Orakel)

- Ein Orakel ist korrekt für einen Test (/ eine Testmenge) wenn $(\forall t \in T :) corr_t(t, p, s) \Rightarrow o(t, p)$.
- Ein Orakel ist vollständig für einen Test (/ eine Testmenge) wenn $(\forall t \in T :) o(t, p) \Rightarrow corr_t(t, p, s)$.

Annahme (Orakelannahme)

1. *Zu jedem Programm p und jeder Spezifiaktion s gibt es ein Orakel o, das vollständig und korrekt ist.*
2. *Zu jedem Programm p und jeder Spezifiaktion s gibt es ein Orakel o, das korrekt ist.*

Einfache Ausweichstrategien

- Nicht verwechseln mit dem falschen Extrapolieren von wenigen vorhandenen Testfällen.
- Ggf. Ausnutzung von Eigenschaften, die ein hinzugezogener Experte kennt.

Beispiel (Staatsverschuldung Deutschlands)

- *Außer EU Bürger: Geldwert.*
- *Deutscher Bürger:* 1 *Billionen* - 10 *Billionen.*
- *Experte:* 2 *Billionen* - $2,5$ *Billionen.*

Einfache Ausweichstrategien

- Nicht verwechseln mit dem falschen Extrapolieren von wenigen vorhandenen Testfällen.
- Ggf. Ausnutzung von Eigenschaften, die ein hinzugezogener Experte kennt.

Beispiel (Sinus)

- $sin^2(x) + cos^2(x) = 1$.
- $-1 \leq sin(x) \leq 1$.
- *Plausibel:* $sin(\pi/4) = 0,34$.
- *Nicht plausibel:* $sin(\pi/5) = -3,2$.

Einfache Ausweichstrategien

- Nicht verwechseln mit dem falschen Extrapolieren von wenigen vorhandenen Testfällen.
- Ggf. Ausnutzung von Eigenschaften, die ein hinzugezogener Experte kennt.

Beispiel (Die 10.000-ste Stelle von π)

- *Alle plausibel:* $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
- *Programm, dass beliebige Stellen ausrechnet.*
- *Prüfen anhand der ersten zehn Stellen.*

Einfache Ausweichstrategien

- Nicht verwechseln mit dem falschen Extrapolieren von wenigen vorhandenen Testfällen.
- Ggf. Ausnutzung von Eigenschaften, die ein hinzugezogener Experte kennt.

Beispiel (Berechnung aller möglichen Bundesligaendtabellen)

- $18! = 6402373705728000$ *viel zu groß.*
- *Counter alleine plausibilisiert schon.*
- *Berechnen für beliebige Anzahlen n von Mannschaften.*

Entwicklung eines Orakels durch maschinelles Lernen

- Unabhängig geschriebenes Programm mit gleichen Spezifikationen.
- Benutze ggf. anderes Programm zum "Trainieren" eines Orakels.
  - Vollautomatisch.
  - Unwahrscheinlich, dass dieselben Fehler auftreten.
- Auch geeignet für Rundungsfehler.
- Ggf. einfach zwei verschiedene Compiler benutzen.

Metamorphisches Testen

- Fachexperte benötigt, aber nur kurz.
- Schöpft Wert aus “wertlosen” Testfällen.
- Sowohl teilweise bekanntes Orakel,
- als auch komplett ohne Orakel.

Metamorphisches Testen - die Methode

- Input-Output-Paar $(x_1, \ldots, x_n, y_1, \ldots, y_m)$ ist korrekt.
- Manipuliere Input nach metamorphischer Regel zu $(x'_1, \ldots, x'_n)$.
- Nach metamorphischer Regel sollte Output $(y'_1, \ldots, y'_m)$ sein.
- Bei n Testfällen und r metamorphischen Regeln $n \cdot 2^r$ neue Testfälle.
- Theoretisch auch ohne korrektes Startpaar möglich.

Metamorphisches Testen - Beispiele

Beispiel (Gauss)

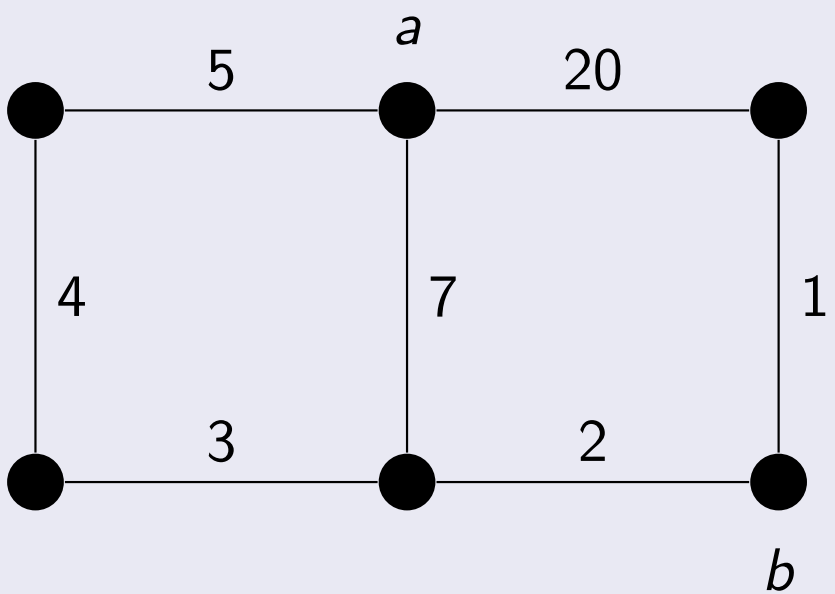
- $A \cdot x = b$
- *Lösung* $x = (x_1, \ldots, x_n)$.
- *Tausche Zeile* i *und* j *von* A *und von* b.
- *Sollte zur gleichen Lösung führen.*
- *Tausche Spalte* i *und* j *von* A.
- *Sollte zur Lösung mit* x_i *und* x_j *vertauscht führen.*

Metamorphisches Testen - Beispiele

Beispiel (Graphen - minimaler Weg)

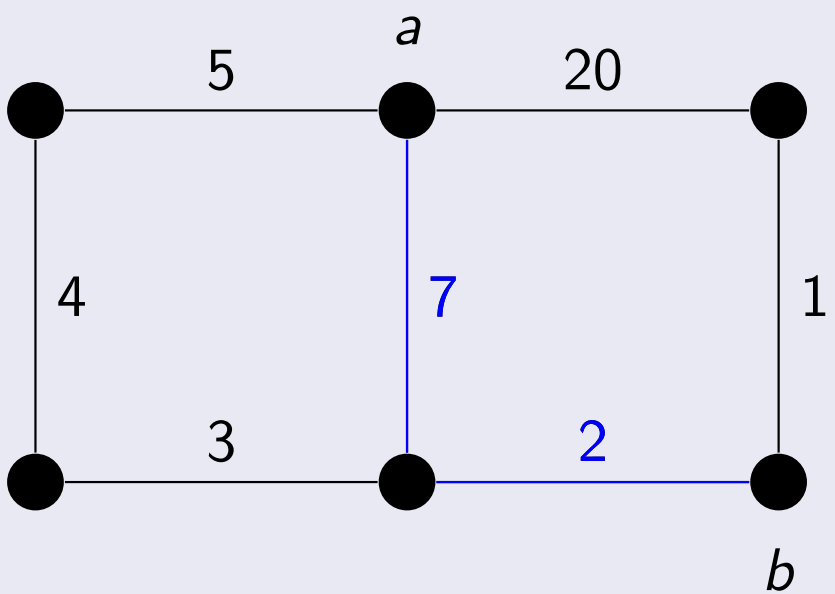
Metamorphisches Testen - Beispiele

Beispiel (Graphen - minimaler Weg)



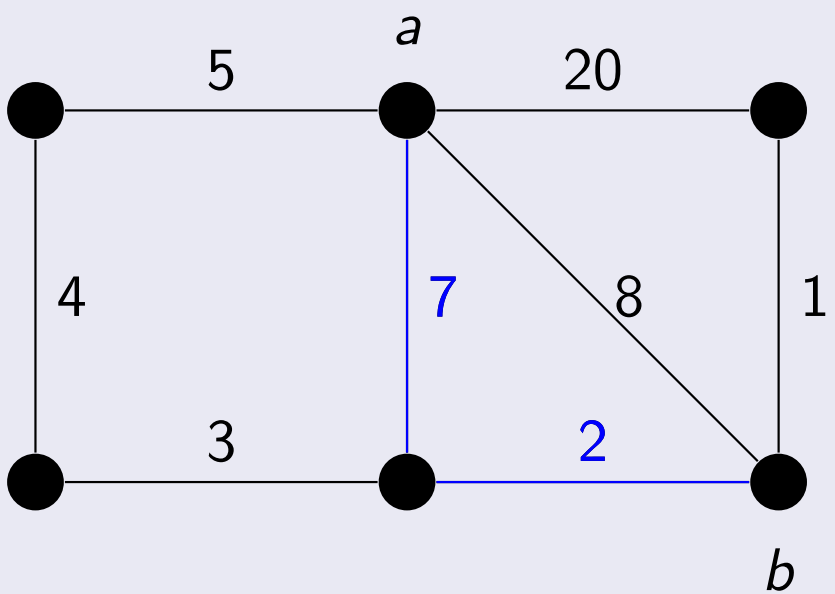
Metamorphisches Testen - Beispiele

Beispiel (Graphen - minimaler Weg)



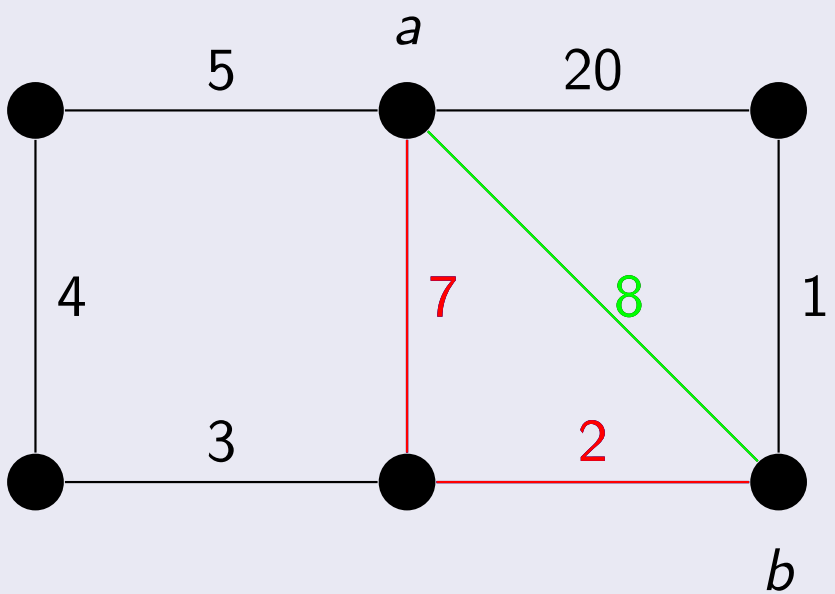
Metamorphisches Testen - Beispiele

Beispiel (Graphen - minimaler Weg)



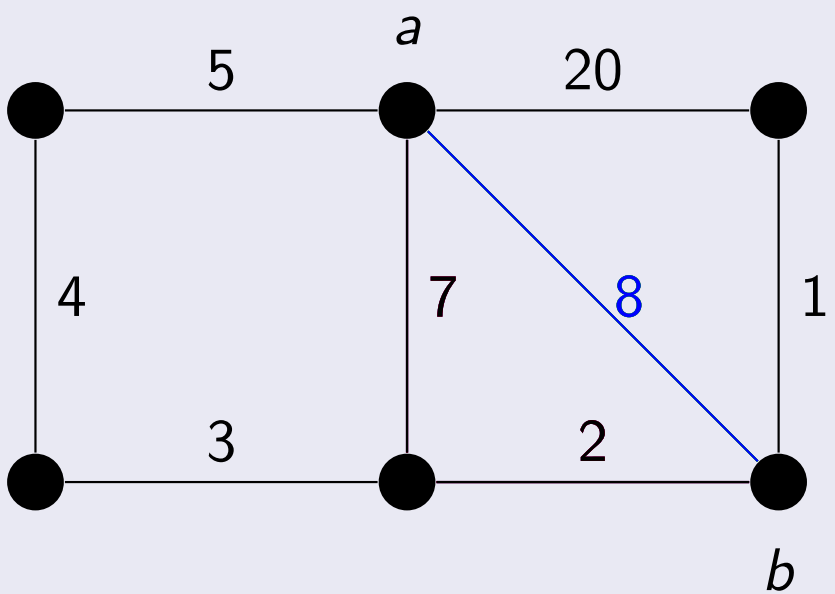
Metamorphisches Testen - Beispiele

Beispiel (Graphen - minimaler Weg)



Metamorphisches Testen - Beispiele

Beispiel (Graphen - minimaler Weg)

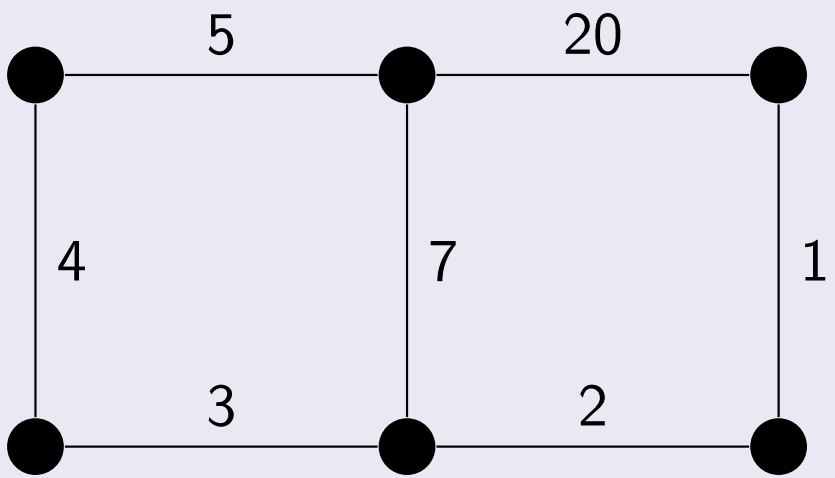


Metamorphisches Testen - Beispiele

Beispiel (Graphen - minimaler Spannbaum)

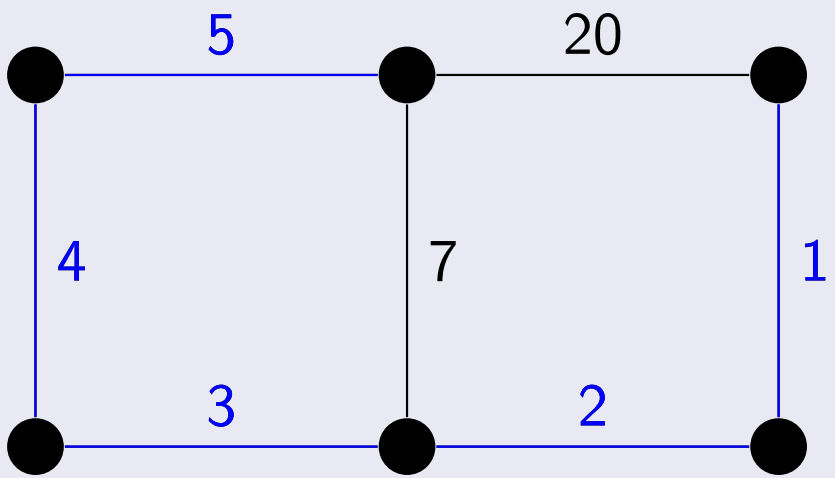
Metamorphisches Testen - Beispiele

Beispiel (Graphen - minimaler Spannbaum)



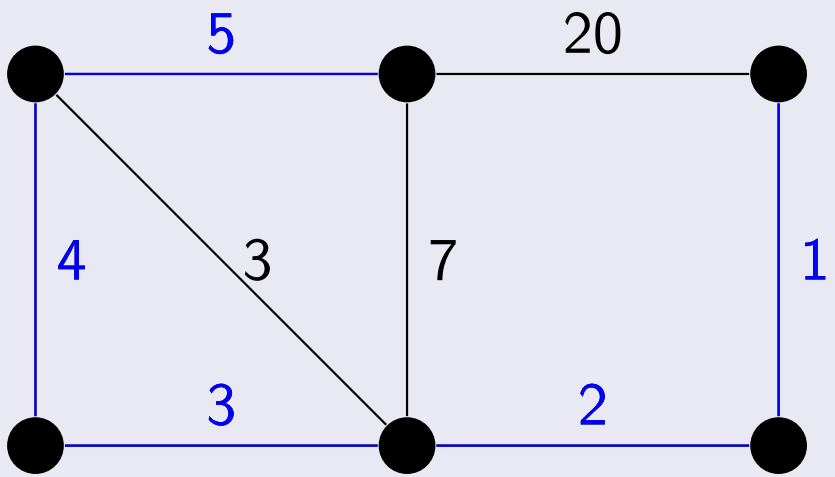
Metamorphisches Testen - Beispiele

Beispiel (Graphen - minimaler Spannbaum)



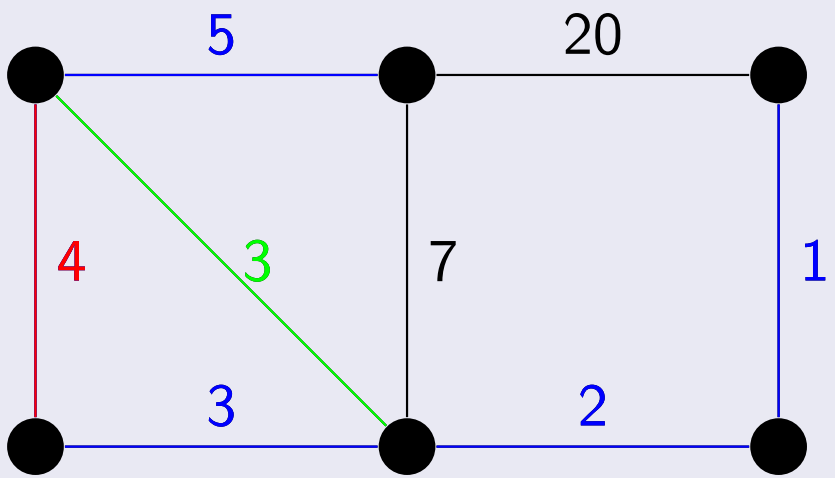
Metamorphisches Testen - Beispiele

Beispiel (Graphen - minimaler Spannbaum)



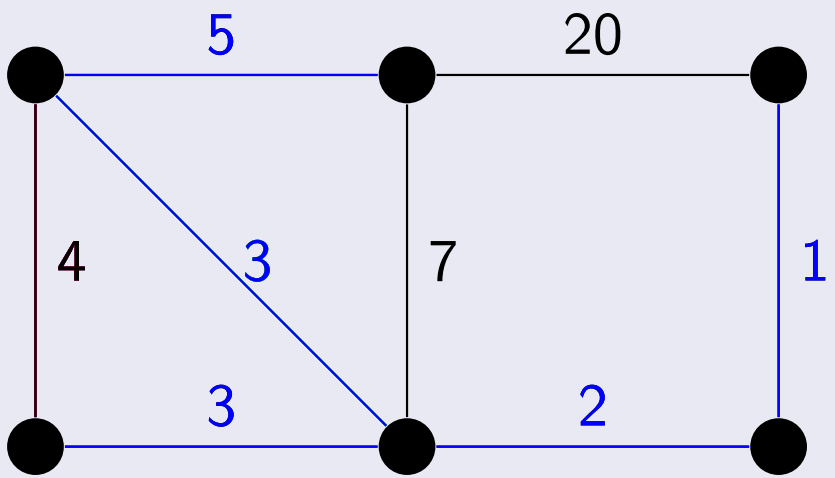
Metamorphisches Testen - Beispiele

Beispiel (Graphen - minimaler Spannbaum)



Metamorphisches Testen - Beispiele

Beispiel (Graphen - minimaler Spannbaum)



Testdaten

- Mit Veröffentlichung des Programms angeben, wie und mit welcher Genauigkeit getestet wurde.
- Im Idealfall Testfälle mitveröffentlichen.

Fazit

- Orakelannahme gilt meist nicht.
- Trotzdem bleibt Testen wichtigstes Fehlerfindewerkzeug.
- Auch “untestbare” Programme mit ein wenig Expertenhilfe sehr wohl testbar.
- Am besten Testdetails mitveröffentlichen.

Literatur

Kanewala, U. and Bieman, J. M. (2013).
Techniques for testing scientific programs without an oracle.
In *2013 5th International Workshop on Software Engineering for Computational Science and Engineering (SE-CSE)*, pages 48–57.

Tsong Y Chen, S. C. and Yiu, S. (1998).
Metamorphic testing: A new approach for generating next test cases.
Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology.

Weyuker, E. J. (1982).
On testing non-testable programs.
Comput. J, 25(4):465–470.