Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Lars Hupel

WS 2015/16
30.10.2015
Abgabe bis: 06.11.2015

Perlen der Informatik
2. Übung

Aufgabe G2.1 Intuitive Berechenbarkeit

Sind die folgenden Funktionen berechenbar? Für welche kann man sogar einen Algorithmus explizit angeben? Dabei sei $g : \mathbb{N} \to \mathbb{N}$ eine totale, berechenbare Funktion und $h : \mathbb{N} \to \mathbb{N}$ eine nicht berechenbare Funktion. Weiter ist über die Funktionen g und h nichts bekannt.

$$f_1(n) = \begin{cases} 1 & \text{falls } n \text{ eine Primzahl ist} \\ 0 & \text{sonst} \end{cases}$$

$$f_2(n) = \begin{cases} 1 & \text{falls es eine Primzahl } p \geq n \text{ gibt} \\ 0 & \text{sonst} \end{cases}$$

$$f_3(n) = \begin{cases} 1 & \text{falls es einen Primzahlzwilling } (p, p+2) \text{ mit } p \geq n \text{ gibt} \\ 0 & \text{sonst} \end{cases}$$

Hinweis: Ein Primzahlzwilling ist ein Paar von Primzahlen, deren Differenz 2 beträgt (z.B. $(11, 13)$). Es ist nicht bekannt, ob es unendlich viele Primzahlzwillinge gibt.

$$f_4(n) = \begin{cases} 1 & \text{falls } g(n) = n \\ 0 & \text{sonst} \end{cases}$$

$$f_5(n) = \begin{cases} 0 & \text{falls } n = 0 \\ h(n) & \text{sonst} \end{cases}$$

$$f_6(n) = \begin{cases} n & \text{falls es ein } m \text{ gibt mit } h(m) \leq m \\ 0 & \text{sonst} \end{cases}$$

Aufgabe G2.2 Konsequenzen des Satzes von Rice

Der Satz von Rice ist ein sehr starkes Unentscheidbarkeitsresultat über Eigenschaften von Programmen. Dennoch gibt es viele Eigenschaften von Programmen, die mit Hilfe von geeigneten Tools geprüft werden können, z.B.

- ob das Programm jemals mehr als k Schritte laufen wird
- ob alle Variablen vor Verwendung initialisiert werden
- ob das Programm typkorrekt ist, also keine Typfehler zur Laufzeit auftreten können

Warum ist die Existenz solcher Verfahren kein Widerspruch zum Satz von Rice? Was hat der Satz von Rice andererseits für Konsequenzen für die Verfahren?

Aufgabe G2.3 Telefonproblem

Wir betrachten eine Variante des Telefon-Problems aus der Vorlesung, mit dem Unterschied, dass die Freunde nun statt Telefonen Briefe benutzen (dass heißt, bei jeder Kommunikation wandert Information nur in eine Richtung).

1. Geben Sie ein Verfahren an, um mit möglichst wenig Briefen die Information an jeden zu verteilen.
2. Zeigen Sie, dass dieses Verfahren die minimale Anzahl Briefe benötigt.

Aufgabe H2.1 Paralleles Telefonproblem

Wir betrachten eine weitere Variante des Telefonproblems: Eine Gruppe von n Freunden geht jeden Samstag zu n verschiedenen Bundesliga-Spielen. Danach rufen sie sich gegenseitig an, um sich die Ergebnisse mitzuteilen.

Es können beliebig viele Gespräche (mit unterschiedlichen Teilnehmern) gleichzeitig stattfinden und jedes Gespräch dauert eine Zeiteinheit, unabhängig von der Anzahl der ausgetauschten Informationen.

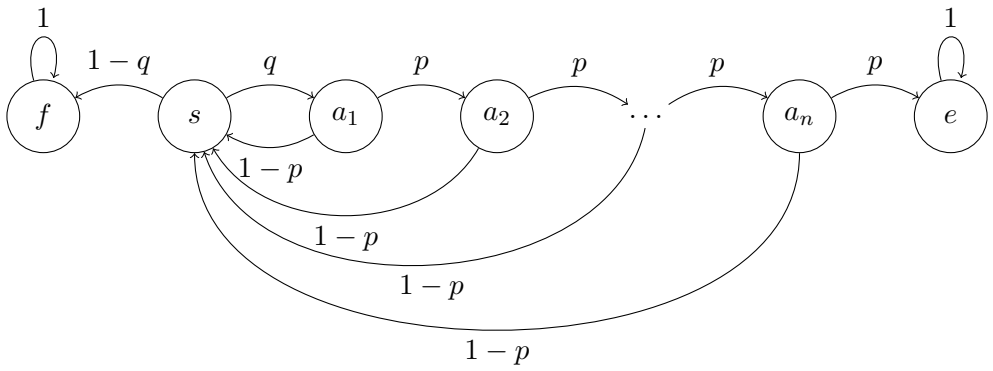
Wie viele Zeiteinheiten sind notwendig, damit jeder am Ende alle Informationen hat?

Aufgabe H2.2 Analyse des Zeroconf-Algorithmus

Der Zeroconf-Algorithmus ist ein Verfahren, mit dem sich Rechner in einem automatisch selbst eine IP-Adresse zuweisen können, ohne dass eine zentrale Vergabestelle benötigt wird oder dass es zu Konflikten kommt.

Der Algorithmus ist dabei der folgende: Ein Rechner, der eine Adresse benötigt, wählt sich aus einem Pool von Adressen zufällig eine freie Adresse aus (mit Zurücklegen). Die Wahrscheinlichkeit, dass es sich um eine bereits belegte Adresse handelt, sei q. Danach fragt der Rechner die anderen Rechner im Netz, ob die Adresse bereits verwendet wurde. Bekommt der Rechner nach n Anfragen keine Antwort, so geht der Algorithmus davon aus, dass kein anderer Rechner die Adresse verwendet und ist fertig. Bekommt der Rechner die Antwort, dass die Adresse bereits verwendet wird, beginnt der Algorithmus von vorn. Die Wahrscheinlichkeit, dass auf eine Anfrage fälschlicherweise keine Antwort kommt, sei p.

Man kann das Problem jetzt als gerichteten Graphen formulieren, dessen Kanten mit Wahrscheinlichkeiten beschriftet sind:



Der Knoten f bezeichnet den Erfolgszustand, s den Startzustand und e den Fehlerzustand. Die Zustände a_i bezeichnen den Zustand nach der i-ten Anfrage. Als Randbedingung gilt, dass $0 < p < 1$ und $0 < q < 1$.

Wie hoch ist die Wahrscheinlichkeit, dass der Algorithmus am Ende eine bereits belegte Adresse zurückgibt (also im Fehlerzustand des obigen Graphen landet)? Der Einfachheit halber darf mit der festen Anzahl von Versuchen $n = 2$ gerechnet werden. Wie lässt sich das Ergebnis auf beliebige n generalisieren?