Statische Analysen von Online-Befragungen mit der Programmiersprache *liQuid*

Thomas M. Prinz, Linda Gräfe, Jan Plötner und Anja Vetterlein

Universitätsprojekt Lehrevaluation
Friedrich-Schiller-Universität Jena
07743 Jena, Deutschland
{Thomas.Prinz, Linda.Graefe, Jan.Ploetner, Anja.Vetterlein}@uni-jena.de

Zusammenfassung Befragungen in der empirischen Forschung und im Qualitätsmanagement werden zunehmend komplexer. Diese wachsende Komplexität entsteht vor allen Dingen durch die steigende Anzahl an Variablen und des sich erhöhenden Verzweigungsgrades in dem zur Befragung gehörenden Fragebogen. Das derzeit genutzte seitenbezogene, informelle Layout zur Beschreibung von Fragebögen wird dieser Komplexität nicht mehr gerecht.
Das Ziel unserer Forschungsarbeiten im Bereich der Psychoinformatik ist die Entwicklung einer domänenspezifischen Sprache zur Beschreibung umfangreicher Fragebögen. Für diese Sprache sollen Analysetechniken zur Verfügung gestellt werden, um die Stimmigkeit (im Sinne der Korrektheit) von Fragebögen zu überprüfen.
Wir schlagen in dieser Arbeit unsere Programmiersprache *liQuid* als domänenspezifische Sprache vor. Diese ermöglicht sowohl eine einfache, technische Beschreibung von Fragebögen als auch die Anwendung von Analysen. Als erstes Beispiel für solch eine Analyse zeigen wir, wie in *liQuid* Mehrfachabfragen ein und derselben Variablen durch statische Analysen erkannt werden können.

1 Einleitung

Befragungen mittels Fragebögen werden in der empirischen Forschung und im Qualitätsmanagement zunehmend komplexer. Die Komplexität vergrößert sich dabei vor allen Dingen in dem zur Befragung gehörenden Fragebogen hinsichtlich des Verzweigungsgrades (der Adaptivität) und der Anzahl an Variablen. Dies geht damit einher, dass immer mehr Befragungen computergestützt durchgeführt werden (bspw. die PISA-Studie[1]). Dadurch eröffnen sich neue Möglichkeiten der Implementierung der Fragebögen. Mit diesen neuen Möglichkeiten erhöht sich auch deren Komplexität. Dies wird auch im Universitätsprojekt Lehrevaluation (ULe) deutlich, in dem einzelne Lehrveranstaltungen sowie ganze Studiengänge an der Friedrich-Schiller-Universität Jena durch Fragebögen evaluiert werden.

Das derzeit in der Fragebogenkonstruktion genutzte, seitenbezogene und informelle Layout zur Beschreibung von Fragebögen wird dieser zunehmenden

[1] http://www.oecd.org/berlin/themen/pisa-studie/ (Juli 2017)

Komplexität kaum mehr gerecht, wie wir bspw. bei Befragungen von ULe feststellen mussten. Zeitgleich werden aber viele Fragebögen in diesem Format als Programme online zur Verfügung gestellt, um Befragungen schnell und einfach durchzuführen (bspw. SoSci Survey[2] und Survey Monkey[3]). Für solche elektronischen, mitunter umfangreichen Fragebögen gibt es derzeit jedoch keine Überprüfung auf deren technische Stimmigkeit, da sie generell eher informeller Natur sind.

Aus diesem Grund forschen wir an Modellen zur Formalisierung von Fragebögen und Methoden um diese Modelle statisch und dynamisch auf ihre technische Stimmigkeit zu überprüfen. Ein vielversprechender Ansatz zum Erreichen dieses Ziels ist die Entwicklung einer domänenspezifischen Programmiersprache zur Beschreibung komplexer Fragebögen. Für diese Sprache möchten wir Analysetechniken zur Verfügung stellen, um die Fragebögen auf Stimmigkeit zu testen. Der Begriff der *Stimmigkeit* ist dabei als Sammelbegriff für verschiedenste Probleme bei der Fragebogenkonstruktion zu verstehen.

In dieser Arbeit führen wir in die für die Fragebogenkonstruktion relevanten Begriffe ein und formalisieren sie, um sie in einem Modell eines Fragebogens vereinen zu können. Dabei verwenden wir nicht mehr ein seitenbezogenes Layout für Fragebögen, sondern greifen auf die Verwendung von Graphen zurück. Dies ermöglicht zum einen die Anwendung etablierter Analysemethoden der Graphentheorie. Zum anderen lassen sich im Gegensatz zum seitenbezogenen Layout durch Graphen die Verzweigungen in den Fragebögen intuitiver modellieren und strukturierter darstellen. Somit wird auch die Struktur eines Fragebogens erstmalig adäquat formalisiert.

Auf Basis dieses Modells führen wir die (konzeptionelle) domänenspezifische Programmiersprache *liQuid* ein. In dieser Arbeit verwenden wir einen eingeschränkten Umfang von *liQuid* bestehend aus verschiedenen Typen (Skalen), Variablen und Items (Fragestellungen). Diese werden genutzt, um Fragebögen zu programmieren. Da ein entstandenes Programm einem Kontrollflussgraphen gleicht, können Analysen aus dem Übersetzerbau erstmals auch auf Fragebögen angewendet werden. Als erstes Beispiel einer solchen Analyse zeigen wir, wie in *liQuid* Mehrfachabfragen einer Variablen erkannt werden können. Dazu werden die aus dem Übersetzerbau bekannten Begriffe der lebendigen Variablen und Datenflussanalysen genutzt.

Dieser Beitrag ist, wie folgt, aufgebaut: In den Grundlagen (Kapitel 2) definieren wir die von uns verwendeten Notationen von Digraphen und Pfaden. Anschließend führen wir Schritt für Schritt Begriffe der Fragebogenkonstruktion ein und formalisieren sie (Kapitel 3). Aus diesen Begriffen wird unsere Definition eines Fragebogens abgeleitet. Das Fragebogenmodell fließt direkt in Kapitel 4 in die Programmiersprache *liQuid* ein. In Programmen dieser Sprache können Mehrfachabfragen von Variablen erkannt werden. Dazu erläutern wir in Kapitel 5 zunächst die Problematik bei solch einer Mehrfachabfragung und wie diese mit Hilfe von lebendigen Variablen und Datenflussanalysen gelöst werden kann. Zum

[2] `https://www.soscisurvey.de/` (Juli 2017)
[3] `https://www.surveymonkey.com/` (Juli 2017)

Schluss fassen wir in Kapitel 6 diesen Beitrag zusammen und geben einen kurzen Ausblick auf zukünftige Arbeiten.

2 Grundlagen

Im Kontext dieser Arbeit ist der Begriff des *gerichteten Graphen* wichtig, um später Strukturen in Form von Graphen zu beschreiben.

Definition 1 (Gerichteter Graph). *Ein* gerichteter Graph *bzw.* Digraph G *setzt sich zusammen aus einer Menge* $N = N(G)$ *an* Knoten *und einer Menge* $E = E(G)$ *an* Kanten *von* G*.* E *bildet dabei eine Menge von geordneten Paaren,* $E \subseteq N \times N$*. Wir schreiben auch* $G = (N, E)$ *[3, S. 432 ff.].*

Durch die Verbindung der Knoten durch Kanten entstehen *Pfade* von einem Knoten zu einem anderen:

Definition 2 (Pfad). *In einem Digraphen* $G = (N, E)$ *heißt eine Sequenz* $P = (n_0, \ldots, n_m)$*,* $m \geq 0$*, von Knoten aus* N*,* $n_0, \ldots, n_m \in N$*, ein* Pfad*, sobald jeder Knoten mit seinem darauffolgenden Knoten durch eine Kante verbunden ist [4, S. 1180]:*

$$\forall\, 0 \leq i < m \colon (n_i, n_{i+1}) \in E \tag{1}$$

Ein Pfad heißt *azyklisch*, wenn alle Knoten des Pfads paarweise verschieden sind. Dem folgend heißt ein Digraph *azyklisch*, wenn jeder Pfad im Graph azyklisch ist. Anderenfalls beinhaltet der Graph eine Schleife und ist zyklisch. Wir beschreiben die Menge aller Pfade von einem Knoten a zu einem Knoten b durch $\mathcal{P}_{a \to b}$. Ein gerichteter Graph heißt *zusammenhängend*, wenn in seinem ungerichteten Pendant[4] von jedem Knoten zu jedem anderen Knoten ein Pfad existiert [10, S. 547].

3 Modell eines Fragebogens

Nachdem die Grundlagen für diesen Beitrag eingeführt wurden, führt dieses Kapitel die wichtigen Begriffe aus dem Bereich der Fragebogenkonstruktion ein. Dabei werden diese, wenn nötig, formalisiert. Anschließend wird aus diesen Begriffen unser Modell eines Fragebogens aufgebaut.

3.1 Variablen und Erhebungen

In der Messtheorie in den Sozial- und Verhaltenswissenschaften geht es um die Untersuchung der Eigenschaften von Objekten, im Speziellen von Menschen [2, S. 8]. In der Realität sind (psychologische) Eigenschaften (z. B. Intelligenz und Motivation) jedoch viel zu komplex und vielgestaltig, sodass diese beim Messen vergröbert und im Kontext der elektronischen Erfassung auch diskretisiert werden.

[4] Im gerichteten Graphen wird für jede Kante (a, b) ebenfalls eine Kante (b, a) eingefügt.

Demzufolge gibt es eine Abbildung von dem tatsächlichen Wertebereich einer Eigenschaft auf eine Menge diskreter Werte. Diese Menge an diskreten Werten nennen wir *Typ*. Dabei sollte eine Übertragung des *empirischen* Relativs auf einen *numerischen* Relativs innerhalb des Typs erfolgen [2, S. 17 ff.].

Durch die vorgenommene Diskretisierung besitzt jede Messung einer Eigenschaft bereits eine Ungenauigkeit. Aber nur mit Hilfe dieser Diskretisierung ist eine Messung überhaupt erst möglich. Somit ist die entstandene Inexaktheit aus praktischer Sicht unvermeidbar. Bei der Auswahl eines geeigneten Typs sollte auf die Reliabilität [9, S. 11] der Messungen geachtet werden.

Messen wir eine Eigenschaft eines Objekts obj mit Hilfe der Werte eines Typs $\mathcal{T}$, dann erhalten wir ein Paar (obj, val) mit $val \in \mathcal{T}$. Dieses Paar sagt aus, dass für obj bezüglich des Typs $\mathcal{T}$ der Wert val gemessen wurde. Führen wir nun Messungen an allen Objekten $\mathcal{O}$ durch, erhalten wir somit eine Abbildung von diesen Objekten auf den Typ. Diese Abbildung nennen wir eine *Variable*.

Definition 3 (Variable). *Eine* Variable *var ist eine linkstotale Abbildung von der Menge aller Objekte $\mathcal{O}$ auf einen Typ $\mathcal{T}$, $var\colon \mathcal{O} \mapsto \mathcal{T}$. Dabei schreiben wir für den Wertebereich der Variablen $\mathcal{T}(var)$.*

Mit einer Variablen werden also die Ausprägungen *einer* Eigenschaft (mit Hilfe einer Diskretisierung) von verschiedenen Objekten beschrieben [11, S. 19]. Die Variablenausprägungen sind vor den Messungen unbekannt. Manchmal sind auch theoretische Annahmen über diese Ausprägungen zugrunde gelegt, sollen jedoch empirisch überprüft werden. In beiden Fällen können wir davon ausgehen, dass sie durch verschiedene Messungen ermittelt werden müssen. Dazu Bedarf es an geeigneten *Messverfahren*. Ein Messverfahren ermittelt für ein Objekt obj und einer Menge von Variablen für jede dieser Variablen var ein Element $(obj, val) \in var$, $value \in \mathcal{T}(var)$. Sind die Variablen im Messverfahren geordnet, so ist das Messverfahren eine Abbildung von der Menge aller Objekte auf das kartesische Produkt der Typen aller im Messverfahren erhobenen Variablen.

Führen wir ein solches Messverfahren auf einer Menge von Objekten durch, dann sprechen wir von einer *Erhebung*.

Definition 4 (Erhebung). *Eine* Erhebung $\mathcal{H}$ *umfasst ein* Messverfahren $\mathcal{M}$, *eine Menge an* Objekten $\mathcal{O}$ *und eine Menge an* Variablen $\mathcal{V}$, $\mathcal{H} = (\mathcal{M}, \mathcal{O}, \mathcal{V})$. *Das Messverfahren $\mathcal{M}$ ist dabei eine Abbildung von der Menge der Objekte $\mathcal{O}$ auf das kartesische Produkt aller Typen aller Variablen:*

$$\mathcal{M}\colon \mathcal{O} \mapsto \prod_{v \in \mathcal{V}} \mathcal{T}(v) \tag{2}$$

Die Ergebnisse $\mathcal{R}$ der Erhebung $\mathcal{H}$ bilden sich aus der Vereinigung aller Ergebnisse aus den Messungen aller Objekte:

$$\mathcal{R} = \bigcup_{o \in \mathcal{O}} \mathcal{M}(o) \tag{3}$$

3.2 Items und Fragebögen

Im letzten Abschnitt haben wir uns allgemein mit den Begriffen befasst, die wichtig sind, um eine Befragung und einen Fragebogen zu definieren. In Hinblick auf Definition 4 ist eine *Befragung* (mittels Fragebogen) also eine spezielle *Erhebung* und der während der Befragung verwendete *Fragebogen* ein Messverfahren [11, S. 18]. Da ein Messverfahren eine Abbildung darstellt, können wir uns einen Fragebogen also einfach als Funktion, Algorithmus oder Programm vorstellen. Technisch erwartet dieses Programm als Eingabe einen Probanden (das Objekt) und führt zu einer Ausgabe, dem Messergebnis.

Wenn ein Fragebogen ein Programm ist, so können wir es auch mit Hilfe einer Programmiersprache beschreiben. Um diese zu entwickeln und bestenfalls Analysen auf diesen Programmen durchführen zu können, müssen wir jedoch einen Fragebogen noch näher betrachten. Nach Rost [11, S. 18] besteht ein Fragebogen aus sogenannten *Items*. Ein Item ist dabei eine konkrete Fragestellung oder Aufforderung inklusive der Antwortmöglichkeiten, die durch eine oder auch mehrere Variablen gegeben sind.

Definition 5 (Item). *Ein* Item $item = (\mathcal{Q}uest, \mathcal{V}ars)$ *besteht aus einer* Fragestellung $\mathcal{Q}uest(item)$ *und einer Menge abgefragter Variablen* $\mathcal{V}ars(item)$.

Manche Items müssen nicht von jedem Probanden abgefragt werden. Dies ist beispielsweise notwendig, wenn die Antwort auf ein Item die Beantwortung anderer Items ausschließt. Demzufolge benötigen wir *Bedingungen*, um Items vor Probanden zu verbergen:

Definition 6 (Bedingung). *Eine* Bedingung *über die Variablen* $v_0, \ldots, v_n$, $n \geq 0$, *ist eine linkstotale Abbildung vom kartesischen Produkt der Wertebereiche der Variablen (der Typen) auf die Menge* $\{wahr, falsch\}$*:*

$$\mathcal{T}(v_0) \times \cdots \times \mathcal{T}(v_n) \mapsto \{wahr, falsch\} \tag{4}$$

Aus der Testtheorie ist bekannt, dass der Aufbau eines Fragebogens Auswirkungen auf die Messergebnisse hat [9, S. 68 ff.]. Aus diesem Grund können wir einen Fragebogen nicht nur mit Hilfe einer Menge von Items und Bedingungen definieren, da Mengen ungeordnet sind und somit der Aufbau des Fragebogens daraus nicht ersichtlich wird. Es hat sich als vielversprechend herausgestellt, den Aufbau eines Fragebogens als gerichteten, azyklischen Graphen zu beschreiben:

Definition 7 (Fragebogengraph). *Ein* Fragebogengraph Q *(kurz* F-Graph*) besteht aus einem azyklischen, zusammenhängenden Digraphen* (I, E) *mit einer Menge von Items* I *und einer Menge von Kanten* E *zwischen den Items. Die im F-Graphen abgefragten Variablen bezeichnen wir mit* $\mathcal{V}ars(Q)$ *und ergeben sich zu* $\bigcup_{i \in I} \mathcal{V}ars(i)$*. Zusätzlich besitzt ein F-Graph noch eine linkstotale Abbildung, Cond, welche jeder Kante* $e \in E$ *eine Bedingung über einer Teilmenge der im F-Graph enthaltenen Variablen* $\mathcal{V}ars(Q)$ *zuweist. Zusammengefasst entspricht ein F-Graph demnach dem Tripel* $(I, E, Cond)$.

Wir nennen einen F-Graphen Q wohlgeformt*, wenn es genau ein Item ohne eingehende und genau ein Item ohne ausgehende Kanten gibt. Dabei heißt das Item ohne eingehende Kanten das* Start- *und das Item ohne ausgehende Kanten das* Enditem*.*

$$Q \text{ ist wohlgeformt} \tag{5}$$

$$\iff$$

$$|\{i \in I(Q) \colon \triangleright i = \emptyset\}| = 1 \wedge |\{i \in I(Q) \colon i \triangleleft = \emptyset\}| = 1 \tag{6}$$

Ohne Beschränkung der Allgemeinheit gehen wir in den nachfolgenden Kapiteln von der Wohlgeformtheit von F-Graphen aus. Im allgemeinen Fall kann durch Hinzufügen eines expliziten Start- und Enditems für jeden F-Graphen seine Wohlgeformtheit sichergestellt werden.

Definition 7 beschreibt den *Aufbau* eines Fragebogens: Den F-Graphen. Auf Basis dieses Aufbaus lässt sich eine Messverfahren definieren, dass die gewünschten Variablen misst. Die Konstruktion dieses Messverfahrens entspricht der Ausführung des durch den F-Graphen beschriebenen Programms. Diese Ausführungssemantik ist jedoch aus Platzgründen nicht Teil der vorliegenden Arbeit.

4 Die konzeptionelle Programmiersprache *liQuid*

Mit Hilfe der im letzten Kapitel eingeführten Definition eines F-Graphen ist es uns nun möglich, konzeptionell eine domänenspezifische Sprache für Fragebögen zu entwerfen: *liQuid*. Aufgrund der im nächsten Kapitel beschriebenen ersten statischen Analyse auf einem Programm dieser Sprache, konzentrieren wir uns an dieser Stelle auf die Konzepte *Type*, *Variable* und *Item*. In Abbildung 1 ist ein Auszug aus der Grammatik des bisherigen Entwurfs von *liQuid* in Form eines Syntaxdiagramms zu sehen.

Das Konzept *Type* dient der Beschreibung von Typen. Ein Typ besitzt einen Bezeichner sowie seine beinhalteten Werte. Dabei werden diskrete Werte als Eingabe erwartet. Diese können entweder in Textform oder als Zahl angegeben werden. Die Definition von Typen in *liQuid* ist über den Befehl *Type* möglich (vgl. Produktionsregel $\langle type \rangle$ aus Abbildung 1).

Eine *Variable* wird, wie in anderen Programmiersprachen üblich, als eine Art Speicherzelle für Werte definiert. Dabei kennzeichnet ein Bezeichner eine Variable. Zusätzlich wird auch noch der Wertebereich der Variablen angegeben, sprich, der Typ. In *liQuid* werden Variablen durch den Befehl *Variable* definiert (vgl. Produktionsregel $\langle var \rangle$ aus Abbildung 1).

Das letzte Konzept *Item* erlaubt die Definition der im F-Graphen zu findenden Fragestellungen und die dabei aufgenommenen Variablen. Weiterhin wird durch das Item auch die Struktur des F-Graphen beschrieben. Dies erfolgt durch die Angabe der unmittelbar folgenden Items.

Zunächst muss jedem Item ein eindeutiger Bezeichner vergeben werden. Auf diesen folgt in geschweiften Klammern die Fragestellung $\langle question \rangle$, eine Kommaseparierte Auflistung von Variablenbezeichnern und eine (möglicherweise leere)

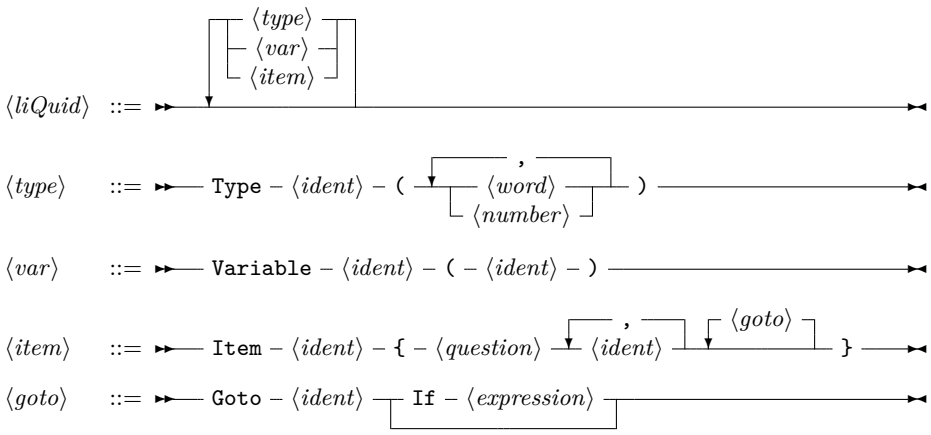

Abbildung 1. Auszug aus der Grammatik von *liQuid*

Auflistung von Sprungbefehlen, ⟨*goto*⟩. Ein Sprungbefehl besteht aus dem Befehl *Goto* und einer Sprungmarke, die einem Itembezeichner entsprechen muss. Optional kann einer Goto-Anweisung eine Bedingung nachgestellt werden. Die Definition eines Sprungbefehls beginnt mit der Angabe von *Goto* (vgl. Produktionsregel ⟨*goto*⟩ aus Abbildung 1). Die Definition eines Items startet mit dem Befehl *Item* (vgl. Regel ⟨*item*⟩ aus Abbildung 1).

Auf die Angabe der Produktionsregeln von ⟨*ident*⟩, ⟨*question*⟩, ⟨*number*⟩, ⟨*word*⟩ und ⟨*expression*⟩ wird aus Gründen der Übersichtlichkeit in Abbildung 1 verzichtet. Ein mittels der Sprache *liQuid* definiertes Beispielprogramm ist in Abbildung 2 zu sehen. Aus diesem Programm lässt sich der F-Graph aus Abbildung 3 gewinnen.

5 Auffinden von mehrfachen Abfragen der selben Variablen

Für die Einführung in die statische Analyse von Fragebögen auf deren (technische) Stimmigkeit untersuchen wir in dieser Arbeit das Auffinden von Mehrfachabfragen ein und derselben Variablen. Dafür müssen wir uns zunächst mit diesem Problem beschäftigen und herausfinden, warum es zu Schwierigkeiten während einer Messung kommen kann.

Nehmen wir dazu den F-Graphen eines *liQuid*-Programms aus Abbildung 3 an. Bei genauerer Betrachtung fällt auf, dass es in diesem Programm einige Variablen, wie die Variablen a und e, gibt, die an mehreren Stellen des F-Graphen abgefragt werden. Im Falle der Variablen e ist dies nicht problematisch, denn ein Proband kann e in jedweder Messung nur genau einmal ausfüllen. Hingegen kann die Variable a während einer Messung mehrmals ausgefüllt werden. Abbildung 4 illustriert solch einen Pfad im Fragebogen, auf dem a mehrmals abgefragt wird.

```
Type s1(1,2,3,4,5)
Type s2(5,4,3,2,1)

Variable a(s1)
Variable b(s2)
Variable c(s2)
Variable d(s1)
Variable e(s2)

Item i1 {"..."
  a
  Goto i2 If a == 4
  Goto i3
}
Item i2 {"..."
  e
  Goto i5 If e == 3
  Goto i6
}
Item i3 {"..."
  b,
  c,
  e
  Goto i4
}
Item i4 {"..."
  d
  Goto i7
}
Item i5 {"..."
  c
  Goto i9
}
Item i6 {"..."
  b
  Goto i7 If e == 2
  Goto i8
}
Item i7 {"..."
  b
  Goto i10
}
Item i8 {"..."
  a
  Goto i9
}
Item i9 {"..."
  d
  Goto i11
}
Item i10 {"..."
  b
  Goto i11
}
Item i11 {"..."
  c
}
```

Abbildung 2. Beispielfragebogen in *liQuid*

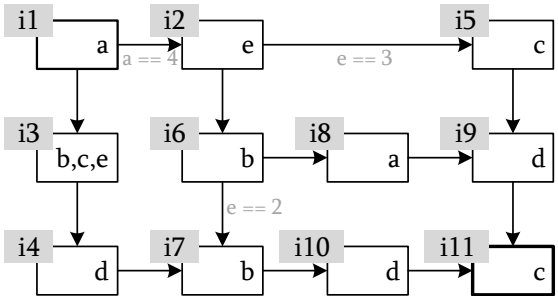

Abbildung 3. Beispiel eines F-Graphen

Solch eine mehrmalige Abfrage von Variablen ist aus den folgenden Gründen eine sowohl technische als auch inhaltliche Schwäche eines *liQuid*-Programms (F-Graphen) und sollte daher ausgeschlossen werden:

1. Bemerkt ein Proband das mehrfache Abfragen derselben Variablen, kann dies zu einer Minderung der Teilnahmemotivation und im schlimmsten Fall zum Abbruch der Befragung führen. Selbst wenn der Proband dies nicht bemerkt, wird der Fragebogen dennoch unnötig verlängert. Dies ist nicht im Sinne der Zumutbarkeit und Unverfälschbarkeit von Fragebögen [9, S. 22 f.].
2. Ein Proband könnte das zur zweiten (oder weiteren) Abfrage gehörende Item anders interpretieren und damit auch anders beantworten. Somit liegen sowohl aus inhaltlicher als auch technischer Sicht zwei unterschiedliche Werte einer Variablen vor. Sei nun beispielsweise angenommen, dass eine vorherige Sprunganweisung auf diese Variable bedingt. Dann kann es vorkommen, dass sich der Proband plötzlich in einem für ihn nicht vorgesehenen Pfad des F-Graphen befindet. Dies ist sowohl für die Interpretation der aufgenommenen Messwerte als auch für eine technische Realisierung ein undefinierter Zustand.

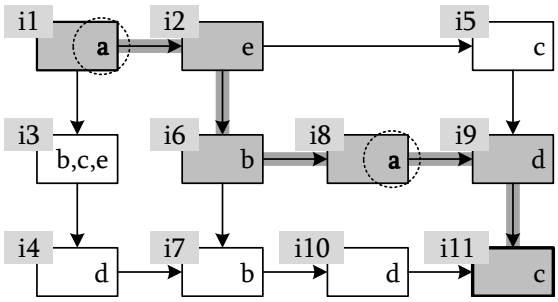

Abbildung 4. Ein Pfad im Fragebogen, in dem die Variable a mehrmals abgefragt wird

Um den zweiten Grund an einem Beispiel zu illustrieren, betrachten wir den F-Graphen aus Abbildung 3. Geht in diesem ein Proband den Pfad $(i1, i2, i6, i8)$ bis zum Item $i8$, dann wissen wir, dass der Proband der Variablen a den Wert 4 zugeordnet haben muss (aufgrund der Bedingung $a == 4$). Wenn er aber nun im Item $i8$ der Variablen a einen Wert *ungleich* 4 zuordnet, befindet er sich plötzlich an einer Position im F-Graphen, an der er nicht sein dürfte.

Aus den genannten Gründen ist die Mehrfachabfrage von Variablen während einer Durchführung einer Messung eines Probanden *nicht* erlaubt. Werden, wie bei Prä-Post-Messungen [6, S. 56 ff.], Variablen von Probanden zu unterschiedlichen Zeitpunkten abgefragt, so sollte dies entweder mittels zweier unterschiedlicher Messungen oder durch die Einführung zweier Variablen für die Prä- und Post-Messung modelliert werden.

Unter diesen Bedingungen ist eine mehrfache Abfrage einer Variablen formal definiert als das Auftreten einer Variablen in mindestens zwei Items, die auf dem selben Pfad von einem Start- zu einem Endknoten liegen. Diese Definition betrachtet eine Mehrfachabfrage aus statischer Sicht und schließt somit den dynamischen Ausschluss von Pfaden mittels nicht-erfüllter Bedingungen aus.

Definition 8 (Mehrfachabfrage von Variablen). *Sei $Q = (I, E, Cond)$ ein wohlgeformter Fragebogen mit dem Startitem s und dem Enditem e, sowie v, $v \in \mathcal{V}ars(Q)$, eine Variable des Fragebogens.*

Q beinhaltet eine Mehrfachabfrage *von v, wenn es einen Pfad vom Start- zum Enditem gibt, auf dem es mindestens zwei verschiedene Items gibt, die v beinhalten.*

$$\exists\, P \in \mathcal{P}_{s \to e}\ \exists\, i_1, i_2 \in P, i_1 \neq i_2 \colon v \in \mathcal{V}ars(i_1) \cap \mathcal{V}ars(i_2) \tag{7}$$

Um nun die Mehrfachabfragen in einem Programm zu erkennen, müssen wir laut Definition 8 alle Pfade vom Start- zum Enditem eines Fragebogens untersuchen. Dies können jedoch im schlechtesten Fall exponentiell viele sein, so dass sich eine Untersuchung auf Basis aller Pfade nicht lohnt und ggf. praktisch nicht möglich ist.

In unseren Untersuchungen haben wir jedoch einen Zusammenhang zwischen der Lebendigkeit von Variablen [8] in normalen Kontrollflussgraphen und dem

Mehrfachabfragen von Variablen in Fragebögen festgestellt. In einem gewöhnlichen Kontrollflussgraphen heißt eine Variable v für einen Knoten n *lebendig*, wenn der Wert von v in einem über n erreichbaren Knoten verwendet wird [8]. Dies ist eine klassische Definition aus der Optimierung zur Eliminierung nicht benötigten Programmcodes [1]. Beispielsweise ist die Variable a des Items $i1$ aus Abbildung 3 im Item $i2$ lebendig, da sie nochmals in Item $i8$ verwendet/definiert wird. Eine Variable v ist also *nur* nach i lebendig, wenn v *nach* i nochmals abgefragt wird. Das heißt, kann festgestellt werden, dass eine Variable v in einem direkt *nach* i folgenden Item *succ* lebendig ist, dann wird v offensichtlich mehrmals abgefragt – es liegt eine Mehrfachabfrage der Variablen v vor.

Um also die Mehrfachabfragen einer Variablen v in einem Fragebogen zu identifizieren, müssen wir lediglich die lebendigen Variablen für jedes Item bestimmen. Dies ist beispielsweise einfach durch eine Datenflussanalyse [5] mit Hilfe der Datenflussgleichung

$$LIVE(i) = DEF(i) \cup \bigcup_{succ \in \{s\,:\,(i,s) \in E\}} LIVE(succ) \tag{8}$$

möglich. Dabei ist $LIVE(x)$ die Menge der lebendigen Variablen des Items x und $DEF(x)$ die Menge der von x abgefragten Variablen. Die Lösung dieser Datenflussgleichung kann in jedem einschlägigen Buch über Übersetzerbau nachgelesen werden (bspw. [1]). Außerdem gehört sie zu den Datenflussproblemen, die sich in linearer Laufzeit für azyklische Graphen lösen lassen [7].

Nehmen wir an, die lebendigen Variablen wurden für jedes Item bestimmt. Anschließend muss für jedes Item i überprüft werden, ob eine von i abgefragte Variable in einem von i's direkten Nachfolgeitems lebendig ist. Ist dies der Fall, so gibt es auf diesem Pfad zum Enditem noch ein weiteres Item, dass diese Variable abfragt. Dann liegt ein Fehler vor. Anderenfalls ist für diese Variable sicher, dass sie zumindest ab diesem Item nicht noch einmal abgefragt wird.

Algorithmus 1 fasst den Ablauf zur Auffindung von Mehrfachabfragen von Variablen zusammen. In Abbildung 5 finden wir unseren Beispielfragebogen, für den die Informationen der lebendigen Variablen abgeleitet wurden. Zur Erinnerung, die lebendigen Variablen werden im inversen Graphen ausgehend vom Enditem berechnet, das heißt, in dem Graphen, in dem jede Kante umgedreht wurde. Wie in der Abbildung zu sehen ist, werden die in ein Item eingehenden Informationen der lebendigen Variablen für das Item vereinigt und durch die vom Item selbst abgefragten Variablen ergänzt (vgl. Gleichung 8).

Wie das Beispiel weiter zeigt, erkennt unser Algorithmus zuverlässig für das Item $i1$, dass die Variable a auf mindestens einem Pfad zum Enditem nochmals abgefragt wird. Dies wird anhand der von Item $i2$ eingehenden lebendigen Variablen $\{a, b, c, d, e\}$ und dem daraus resultierenden, nochmaligen Abfragen von a im Anschluss von $i1$ erkannt. Das gleiche gilt für die Variable b im Item $i6$, denn über den unteren Pfad wird die Variable b bereits abgefragt. Für die Variable e wird kein Fehler festgestellt, da es keinen Pfad vom Item $i1$ zum Item $i11$ gibt, auf dem e mehrfach abgefragt wird. Dies erkennt der Algorithmus korrekt.

Algorithm 1 Bestimmung von Mehrfachabfragen aller Variablen

Input: Fragebogen $Q = (I, E, Cond)$.
Output: Alle Mehrfachabfragen als Menge $Multiple \subseteq I \times \mathcal{V}ars(Q)$.

```
/** Berechne Datenflusssystem für lebendige Variablen.
    LIVE(i) ⊆ Vars(Q) sei die Menge der lebendigen Variablen für das Item i. **/
liveVariablesAnalysis(I, E)
/** Bestimme Mehrfachabfragen der Variablen für alle Items. **/
for all i ∈ I do
    for all succ ∈ {s: (i, s) ∈ I} do
        /** Bestimme Variablen, die im aktuellen Item i abgefragt werden und
            im direkten Nachfolger succ lebendig sind. **/
        alreadyDefined ← LIVE(succ) ∩ Vars(i)
        for all var ∈ alreadyDefined do
            /** Diese sind fehlerhaft. **/
            Multiple ← Multiple ∪ {(i, var)}
```

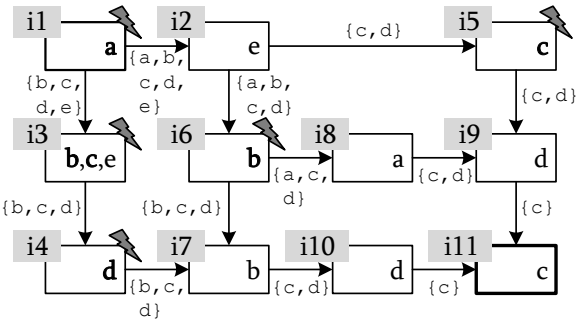

Abbildung 5. Statische Analyse des Beispielfragebogens

Das asymptotische Laufzeitverhalten des Algorithmus wird durch zwei Schritte bestimmt: (1) Die Bestimmung der lebendigen Variablen und (2) die Bestimmung der mehrfach definierten Variablen für jedes Item. Wir bereits erwähnt, zählt die Bestimmung der lebendigen Variablen mittels Datenflussanalyse (1) zu einem der einfachen Datenflussprobleme und ist in azyklischen Graphen in linearer Laufzeit $O(E)$ in Abhängigkeit der Kantenmenge möglich [7]. Die Bestimmung der mehrfach definierten Variablen (2) wird durch 3 Schleifen bestimmt: Die äußere, die mittlere und die innere For-All-Schleife. Die äußere und mittlere For-All-Schleife iterieren zusammen genau einmal über alle Kanten des Fragebogens. Demzufolge entspricht deren gemeinsames asymptotisches Laufzeitenverhalten $O(E)$. Die innere For-All-Schleife hingegen kann jedes Mal im Worst-Case über jede Variable des Fragebogens Q iterieren, $O(\mathcal{V}ars(Q))$. Es ergibt sich demnach eine Laufzeit von $O(E + \mathcal{V}ars(Q) * E)$ und demzufolge $O(\mathcal{V}ars(Q) * E)$.

6 Zusammenfassung

In diesem Beitrag führten wir zunächst wichtige Begriffe aus der Fragebogenkonstruktion für diese Arbeit ein. Daraus entwickelten wir ein Modell eines Fragebogengraphen, das einem azyklischen Digraphen entspricht und aus Items, Kanten und Kantenbedingungen besteht. Jedes Item umfasst eine Menge von Variablen, die durch das Item abgefragt werden.

Auf Basis dieses Fragebogenmodells entwickelten wir eine konzeptionelle Programmiersprache *liQuid*. Mit dieser können Fragebögen definiert, analysiert und ausgeführt werden. Als Beispiel einer Analyse wurde das Problem der Mehrfachabfrage von Variablen eingeführt und gezeigt, wie dieses mit Hilfe einer statischen Analyse mittels lebendiger Variablen gelöst werden kann.

In zukünftigen Arbeiten wird im Universitätsprojekt Lehrevaluation die Sprache *liQuid* um weitere Sprachkonstrukte, wie Repräsentationen und Regeln, ergänzt. Zudem soll die Semantik der Fragebogengraphen formalisiert sowie andere (technische) Probleme bei der Konstruktion von Fragebögen, wie die Nutzung von Variablen in Bedingungen vor deren Abfragung, analysiert und gelöst werden.

Literatur

1. Aho, A.V., Lam, M.S., Sethi, R., Ullman, J.D.: Compiler: Prinzipien, Techniken und Werkzeuge. 2., aktualisierte Auflage, Pearson, München, Deutschland (2008)
2. Bühner, M., Ziegler, M.: Statistik für Psychologen und Sozialwissenschaftler. 1. Auflage, Pearson, München, Deutschland (2009)
3. Chartrand, G., Zhang, P.: Discrete Mathematics. 1 edn., Waveland Press, Inc., Long Grove, Illinois, USA (2011)
4. Cormen, T.H., Leiserson, C.E., Rivest, R., Stein, C.: Algorithmen - Eine Einführung. 4., durchgesehene und korrigierte Auflage, Oldenbourg, Munich, Germany (2013)
5. Hecht, M.S.: Flow Analysis of Computer Programs. Programming Languages Series, Vol. 5, 1 edn., Original von University of California, Elsevier North Holland, New York (1977)
6. Holling, H., Schmitz, B. (eds.): Handbuch Statistik, Methoden und Evaluation. Handbuch der Psychologie, Band 13. 1. Auflage, Hogrefe, Göttingen, Germany (2009)
7. Kam, J.B., Ullman, J.D.: Global Data Flow Analysis and Iterative Algorithms. Journal of the ACM 23(1), pp. 158–171 (Jan 1976)
8. Kennedy, K.: A Global Flow Analysis Algorithm. International Journal of Computer Mathematics 3(1-4), pp. 5–15 (1972)
9. Moosbrugger, H., Kelava, A. (eds.): Testtheorie und Fragebogenkonstruktion. 2., aktualisierte und überarbeitete Auflage, Springer, Berlin, Germany (2011)
10. Pahl, P.J., Damrath, R.: Mathematical Foundations of Computational Engineering: A Handbook. 1. Auflage, Springer, Berlin, Germany (2001)
11. Rost, J.: Lehrbuch Testtheorie – Testkonstruktion. Zweite, vollständig überarbeitete und erweiterte Auflage, Huber, Bern, Schweiz (2004)