Software Modeling
and Verification Chair

RWTH AACHEN
UNIVERSITY

— Bachelor's or Master's Thesis —

Distribution Transformers for Probabilistic Programs

What is it all about?

Probabilistic programs extend deterministic programs by a random choice about which code branch is executed next. They can be defined by the following grammar:

$$\mathrm{c} \coloneqq \mathrm{skip} \mid x \coloneqq \mathrm{a} \mid \{\mathrm{c}\}[\mathrm{p}]\{\mathrm{c}\} \mid \mathrm{c};\mathrm{c} \mid \mathrm{if\ b\ then\ c\ else\ c\ end} \mid \mathrm{while\ b\ do\ c\ end}.$$

Distributions over program states describe what values the program variables hold at a certain time, and with which probability. Our goal is to reason about distributions as pre- and postconditions of probabilistic programs. Consider the following two examples:

$$
\begin{aligned}
&// \; 1 : x \mapsto 1 \\
&\qquad x := x - 1; [\tfrac{1}{2}] x := x + 1; \\
&// \; \tfrac{1}{2} : x \mapsto 0 \text{ and } \tfrac{1}{2} : x \mapsto 1
\end{aligned}
$$

$$
\begin{aligned}
&// \; 1 : x \geq 42 \\
&\qquad x := 0; \\
&// \; 1 : x \mapsto 0
\end{aligned}
$$

A function mapping an initial distribution to the post distribution that the probabilistic program outputs (or vice-versa!) is called a distribution transformer. One such transformer is the denotational semantics of probabilistic programs (see e.g. [dH99]). Currently, I am working on developing more such transformers that vary in their use case (single distributions or sets of distributions) as well as their direction (forwards or backwards). I will present you more details in our initial meeting. Some of those distribution transformers have the interesting property, that they are hard to compute or not defineable in a purely inductive manner.

What is to be done?

The goals of this project are:

1. Understand and apply a variety of distribution transformers to multiple examples
2. Explore requirements under which the distribution transformers are a) computable or b) defineable purely by induction
3. Find easy and nice-looking characterizations/definitions of the transformers

This list is of course non-exhaustive! The above suggestions may be changed, shortened and/or extended while we work on our project and gain more insights on how difficult the topic is.

What we expect:

- Solid background in theoretical computer science and maths – ideally you have already taken theoretical CS electives
- Passion and endurance for solving theoretical problems

What you can expect:

- Get a chance to work on relevant problems of both theoretical and practical nature
- You can work in the student room at our chair – we have a coffee machine, lots of tea and sometimes cookies :)

Apply

- Daniel Zilken (daniel.zilken@cs.rwth-aachen.de)
  Please introduce yourself briefly and say why you're interested in this topic!

References

[dH99] J. I. den Hartog. Verifying probabilistic programs using a hoare like logic. In P. S. Thiagarajan and Roland Yap, editors, *Advances in Computing Science — ASIAN'99*, pages 113–125, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.