RHEINISCH-
WESTFÄLISCHE
TECHNISCHE
HOCHSCHULE
AACHEN

LEHRSTUHL FÜR
INFORMATIK 2
SOFTWARE MODELLIERUNG
UND VERIFIKATION
(MOVES)

— Master Thesis —

Invariants for Conditional Weakest–Preexpectations

What's it all about?

Conditioning is an important, yet not so well–understood, feature in today's probabilistic programming languages. Conditioning is realized by inserting so called observations into the program. An observation acts similar to an assertion—i.e. it blocks undesired program executions—but renormalizes the total probability mass of the remaining (desired) program runs to 1.

```
{$x := 5$} [$1/2$] {$x := 4$};
if $x < 5$ {
    {$y := 1$} [$1/2$] {$y := 2$}
} else {
    $y := 3$
}; observe $x + y \geq 6$
```

Figure 1: Example program for conditioning.

As an example consider the program in Figure 1. The program first flips a fair coin and depending on the outcome either sets x to 5 or to 4. If x was set to 5 it again flips a fair coin and depending on the outcome either sets y to 1 or to 2. Otherwise it deterministically sets y to 3. Finally, the conditioning `observe` $x+y \geq 6$ is performed.

What is now the probability that y is equal to 3 after executing the program? *Without* the observation at the end it would be $1/2$. *With* the statement `observe` $x + y \geq 6$, however, it is $2/3$. The reason is that the run that does not satisfy the condition $x + y \geq 6$ (i.e. the run with probability $1/4$ in which x is set to 4 and y is set to 1) is not taken into consideration but rather the conditional probability of $y = 3$ given $x + y \geq 6$ is of interest. This conditional probability is given by $\frac{1}{2}/1-\frac{1}{4} = 2/3$.

Recently, we have developed cwp—a weakest precondition style semantics to give a reasonable formal meaning to probabilistic programs with observations. Even programs with potentially unbounded while loops are captured properly by means of cwp. As the semantics of a while loop is usually defined in terms of a least fixed point, some sort of calculus is necessary to reason about the meaning of a while loop. While for deterministic programs and probabilistic programs *without* conditioning such a calculus already exists (cf. Hoare–calculus, probabilistic Hoare–calculus, etc.), it is still lacking for such programs *with* observations.

What is to be done?

The goal of this master thesis is to develop proof rules based on invariants to reason about probabilistic programs with observations. In particular an invariant based proof rule for partial correctness as well as a proof rule for total correctness is to be explored.

What should I bring to the table

Apart from general interest in theoretical computer science some background in semantics is helpful but *not required*.

Kontakt

Benjamin Kaminski, Raum 4230, `benjamin.kaminski@informatik.rwth-aachen.de`, Tel. 0241 / 80-21208.