Probability Generating Function Semantics for Probabilistic Programs

by
Clara Elisabeth Scherbaum

Bachelor Thesis at RWTH Aachen University,
Lehrstuhl für Informatik 2

First Examiner:
Prof. Dr. Ir. Joost-Pieter Katoen
Second Examiner:
Prof. Dr. Thomas Noll
Thesis Adviser:
Benjamin Kaminski

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe.

Aachen, November 3, 2017

Clara Scherbaum

Abstract

This thesis presents the definition of a semantics for probabilistic program language based on probability generating functions (PGFs). A semantics of a program is the meaning of that program and its precise description can be useful for verifying properties of a program like termination [**?**]. The semantic rules are defined. Additionally, they are analysed regarding the questions of whether they are well-defined and linear. Those properties are fulfilled. PGFs allow a compact representation of a probability distribution of the values of the program variables after the execution of a program. Other stochastic measurements such as expected values can be extracted quite easily given a PGF. Obtaining a PGF resulting from a program instruction with the semantic rules is most challenging for while-loops. Therefore, we develop two proof rules which simplify this. After that, we show that these proof rules can be applied to a while-loop of a certain form. This form allows the while-loop to contain a probabilistic decision based on a set of different probabilities, namely to permute variables and to sum up variables with constants.

Contents

Chapter 1

Introduction

Software is used in a variety of fields which makes comprehensive verification of software a crucial topic in computer science. One approach to software verification is to test it on critical inputs. However, it is difficult to define what a critical input is. In addition, tests cannot guarantee the absence of bugs. Therefore, another approach to software verification is to do it formally with mathematical methods [**?**]. Such methods can guarantee that the program does what it is supposed to do. One idea on how to formally verify a program is based on formalising the semantics of a program. The semantics of a program is its meaning. Formalizing the semantics of a program has the advantage that we have a precise description of what the program does, regardless of the machine it is executed on. This description can be checked in order to verify that the program behaves correctly [**?**]. In this bachelor thesis, we study a novel approach for a semantics for probabilistic programs. Since we look at probabilistic programs we are interested in the probability distribution of the variables after a program is executed. In addition, it is interesting to know the probability of termination of a program. With our semantics, the probability of termination and the probability distribution of the variables can be computed easily for all program instructions except for loops. For a while-loop, we have to do a more complex analysis to obtain the probability distribution of the variables after its execution. In contrast to the computation of the semantics of the other instructions a program can have (assignments etc.), this analysis cannot always be done by a computer. This makes loops difficult to verify. We try to develop proof rules which allow us to simplify this analysis in certain cases such that it can be done at least partially by a computer.

The semantic rules we define are based on probability generating functions (PGFs). PGFs are one possibility for describing a probability distribution. They give us the chance to extract stochastic measurements such as the expected value of a variable after the execution of the program. This extraction can be done quite easily compared to

other descriptions of probability distributions such as probability mass functions. The first chapter is devoted to PGFs and how we can extract stochastic measurements from them. The second chapter gives an overview of semantics and we define the rules of our semantics. Since it is quite difficult to compute the result of a while-loop, in Chapter 3 we propose some proof rules which let us deal with while-loops of a certain form in an efficient way.

Chapter 2

Preliminaries

In this bachelor thesis we formalise the semantics of a probabilistic program. With that, we want to analyse probabilistic programs. To do this we have to deal a lot with probability distributions. One way to represent a probability distribution is as a formal power series. This has the advantage that we can easily manipulate those series and obtain stochastic measurements like the expected value or the variance [**?**, **?**]. Therefore, this chapter has three parts: One is about formal power series, one about PGFs and the last part is about semantics in general and the underlying theory of domains.

2.1 Formal Power Series

To understand what a formal power series is, we start a few steps earlier and explore the properties of a polynomial. A polynomial is a sum of a number of terms, each made up of a variable raised to some power multiplied by some coefficient.

Definition 2.1.1 (Polynomial of Degree n [**?**])**.**
A polynomial a of degree n is an expression of the form:

$$a = \sum_{i=0}^{n} a_i X^i \qquad \text{with } a_n \neq 0.$$

△

Normally X is an unknown variable and $a_i \in \mathbb{R}$ [**?**]. A polynomial can be represented by the sequence of its coefficients $(a_0, ..., a_k, 0, ...)$. This representation is unique [**?**].

In contrast to polynomials formal power series are basically polynomials with an infinite degree [**?**].

Definition 2.1.2 (Formal Power Series)**.**
A formal power series a is an expression of the form [**?**]:

$$a = \sum_{i \in \mathbb{N}} a_i X^i$$

△

Like polynomials, any formal power series can be uniquely represented by the sequence of its coefficients $(a_0, a_1, ...)$ [**?**]. We can now ask ourselves what the *formal* in *formal power series* means. The *power series* is obvious: we have a series where the power of X is different for every term. The *formal* is meant to emphasise that we treat those power series as algebraic objects and not as analytic objects. If we look at a power series from an analytic point of view we are interested in convergence of this series for specific values of X. For any value of X the power series converges or diverges and we have to take that into account when carrying out operations. However, when looking at *formal* power series we take the power series as algebraic objects and the question of convergence is subordinate even when carrying out operations [**?**].

In order to specify that we look at a certain coefficient of a formal power series, we use the following notation:
Definition 2.1.3 (Referring to Certain Coefficients of a Formal Power Series)**.**
Let $a = \sum_{i \in \mathbb{N}} a_i X^i$ be an formal power series. Then for all $i \in \mathbb{N}$, the following notation is used

$$\left[X^i\right] a = a_i.$$

Operations on Formal Power Series

We can carry out several operations on formal power series like addition, subtraction, scalar-multiplication and derivation [**?**, **?**]. All the operations work very similar to applying those operations on polynomials.

- **Addition/Subtraction:** Addition and subtraction is done coefficient-wise [**?**].

$$\sum_{i \in \mathbb{N}} a_i X^i \pm \sum_{i \in \mathbb{N}} b_i X^i = \sum_{i \in \mathbb{N}} \left(a_i \pm b_i\right) X^i$$

- **Multiplication:** The multiplication is done by the Cauchy product rule [**?**].

$$\sum_{i \in \mathbb{N}} a_i X^i \cdot \sum_{i \in \mathbb{N}} b_i X^i = \sum_{i \in \mathbb{N}} c_i X^i \qquad \text{with } c_i = \sum_{n=0}^{i} a_n b_{i-n}$$

In particular:

$$\begin{aligned} c \cdot \sum_{i\in\mathbb{N}} a_i X^i &= \left(c + \sum_{i\geq 1} 0 \cdot X^i\right) \cdot \sum_{i\in\mathbb{N}} a_i X^i \\ &= \sum_{i\in\mathbb{N}} c \cdot a_i X^i, \text{ and} \\ X^m \cdot \sum_{i\in\mathbb{N}} a_i X^i &= \left(X^m + \sum_{i\in\mathbb{N}\setminus\{m\}} 0X^i\right) \cdot \sum_{i\in\mathbb{N}} a_i X^i \\ &= \sum_{i\in\mathbb{N}} a_i X^{i+m}, \qquad \text{for all } m \geq 0 \qquad [?] \end{aligned}$$

- **Derivation:** The derivation of a polynomial is obtained by reducing all exponents by one and multiplying the corresponding coefficient with the original exponent. Exactly the same is done for formal power series and we get the following formal power series for the first and second derivation [?, ?]. Suppose $f(X) = \sum_{i\in\mathbb{N}} a_i X^i$ is a formal power series.

$$\begin{aligned} \frac{\partial f}{\partial X} &= \sum_{i\in\mathbb{N}} (i+1) a_{i+1} X^i = \sum_{i\in\mathbb{N}} i a_i X^{i-1} \\ \frac{\partial^2 f}{\partial X^2} &= \sum_{i\in\mathbb{N}} i(i+1) a_{i+2} X^i = \sum_{i\in\mathbb{N}} i(i-1) a_i X^{i-2} \end{aligned}$$

  We can ask ourselves why the second notion with X^{i-1} and X^{i-2} respectively is valid since we defined formal power series such that the powers are always positive. In fact, all terms where the power of X is less than 0 are zero since i or $i-1$ are zero in that case.
  In general, the notation $\frac{\partial^i f}{\partial X^i}$ denotes the i-th (partial) derivation with respect to X.

Multivariate Formal Power Series

There exist polynomials with more than just one unknown variable, for example we can define a polynomial $\sum_{0\leq i_1,\ldots,i_k\leq n} a_{i_1,\ldots,i_k} X_1^{i_1} \cdots X_k^{i_k}$ [?]. This concept can be transferred to the theory of formal power series and we can define a formal power series with multiple unknown variables [?].

Definition 2.1.4 (Multivariate Formal Power Series)**.**
A formal power series a with an arbitrary number of unknown variables is an expression

of the form [**?**]:

$$a = \sum_{i_1,...,i_k \in \mathbb{N}} a_{i_1,...,i_k} X_1^{i_1} \cdots X_k^{i_k}$$

△

With that definition in mind, we can define all the operations we have defined for formal power series with exactly one unknown for the multivariate case.

Let $f(X_1, ..., X_k)$ be the formal power series $\sum_{i_1,...,i_k \in \mathbb{N}} a_{i_1,...,i_k} X_1^{i_1} \cdots X_k^{i_k}$.

- **Addition/Subtraction** [**?**]:

$$\begin{aligned} &\sum_{i_1,...,i_k \in \mathbb{N}} a_{i_1,...,i_k} X_1^{i_1} \cdots X_k^{i_k} \pm \sum_{i_1,...,i_k \in \mathbb{N}} b_{i_1,...,i_k} X_1^{i_1} \cdots X_k^{i_k} \\ = &\sum_{i_1,...,i_k \in \mathbb{N}} \left(a_{i_1,...,i_k} \pm b_{i_1,...,i_k}\right) X_1^{i_1} \cdots X_k^{i_k} \end{aligned}$$

- **Multiplication:** Multiplication is done by the Cauchy product rule [**?**].

$$\begin{aligned} &\left(\sum_{i_1,...,i_k \in \mathbb{N}} a_{i_1,...,i_k} X_1^{i_1} \cdots X_k^{i_k}\right) \cdot \left(\sum_{i_1,...,i_k \in \mathbb{N}} b_{i_1,...,i_k} X_1^{i_1} \cdots X_k^{i_k}\right) \\ = &\sum_{i_1,...,i_k \in \mathbb{N}} c_{i_1,...,i_k} X_1^{i_1} \cdots X_k^{i_k} \\ &\text{with } c_{i_1,...,i_k} = \sum_{\substack{n_1,...,n_k \in \mathbb{N} \wedge \\ n_j \leq i_j \text{ for all } j \in \{1,...,k\}}} a_{n_1,...,n_k} \cdot b_{i_1 - n_1,...,i_k - n_k} \end{aligned}$$

  In particular:

$$\begin{aligned} &c \cdot \sum_{i_1,...,i_k \in \mathbb{N}} a_{i_1,...,i_k} X_1^{i_1} \cdots X_k^{i_k} \\ = &\left(c + \sum_{i_1,...,i_k \in \mathbb{N} \setminus \{0\}} 0 X_1^{i_1} \cdots X_k^{i_k}\right) \cdot \sum_{i_1,...,i_k \in \mathbb{N}} a_{i_1,...,i_k} X_1^{i_1} \cdots X_k^{i_k} \\ = &\sum_{i_1,...,i_k \in \mathbb{N}} c \cdot a_{i_1,...,i_k} X_1^{i_1} \cdots X_k^{i_k}, \text{ and} \\ &X_j^m \cdot \sum_{i_1,...,i_k \in \mathbb{N}} a_{i_1,...,i_k} X_1^{i_1} \cdots X_k^{i_k} \\ = &\left(X_1^0 \cdots X_j^m \cdots X_k^0 + \sum_{\substack{i_1,...,i_k \in \mathbb{N} \wedge \\ (i_1,...,i_k) \neq (0,...,m,...,0)}} 0 X_1^{i_1} \cdots X_k^{i_k}\right) \end{aligned}$$

$$\cdot \sum_{i_1,\ldots,i_k \in \mathbb{N}} a_{i_1,\ldots,i_k} X_1^{i_1} \cdots X_k^{i_k}$$
$$= \sum_{i_1,\ldots,i_k \in \mathbb{N}} a_{i_1,\ldots,i_k} X_1^{i_1} \cdots X_j^{i_j+m} \cdots X_k^{i_k} \qquad \text{for all } m \geq 0$$

- **Derivation:** In contrast to univariate formal power series, with several unknown variables we can compute partial derivations with respect to any of the unknown variables [**?**].

$$\frac{\partial f}{\partial X_j} = \sum_{i_1,\ldots,i_k \in \mathbb{N}} i_j a_{i_1,\ldots,i_k} X_1^{i_1} \cdots X_j^{i_j-1} \cdots X_k^{i_k}$$

Convergence

Although we said at the beginning that the question of convergence of a specific formal power series is secondary when carrying out operations, it is still interesting to think about convergence. There are some formal power series for which a closed form exists [**?**]. The most important for this bachelor thesis is the *geometric series* which converges for $|Z| < 1$ or if we do not insert any values for the unknown variables [**?**, **?**]:

$$\sum_{i \in \mathbb{N}} Z^i = \frac{1}{1-Z}$$

Of course, Z can be a product of unknown variables and coefficients.

2.2 Probability Generating Functions

There exists an method for combining the concepts of probability distributions and formal power series which we will now introduce [**?**].

Generating Functions

In the last section, we noticed that any formal power series could be uniquely represented by the sequence of its coefficients. In fact the converse is also true: All sequences of numbers can be represented by a formal power series [**?**]. Let $\{a_i\}_{i \in \mathbb{N}}$ be a sequence. Then this sequence can be represented by the formal power series:

$$\sum_{i \in \mathbb{N}} a_i X^i.$$

This formal power series is then called the *generating function* of the original sequence [**?**].

Of course, the number and name of unknown variables of this formal power series can differ depending on what representation is convenient for our application (as done in [**?**]).

Probability Generating Functions

Normally we represent the probability distribution of a discrete random variable $\mathtt{X}$ as a probability mass function [**?**, **?**]:

$$\mu\colon \mathbb{N} \to [0,1],\ i \mapsto \Pr(\mathtt{X} = i)$$

All the probabilities sum up to a value less than or equal to one [**?**]. So

$$\sum_{i \in \mathbb{N}} \mu(i) \leq 1.$$

We can define a sequence based on the probability mass function:

$$\{\mu(i)\}_{i \in \mathbb{N}}$$

This sequence – the probability distribution – can also be represented as a generating function: the *probability generating function (PGF).*

$$G_\mu = \sum_{i \in \mathbb{N}} \mu(i) X^i$$

We have to be aware of the fact that we have two different symbols X and $\mathtt{X}$ with two different meanings. On the one hand, $\mathtt{X}$ is a random variable which can be any natural number. On the other hand, X is the unknown variable of the PGF. As such it can be set to any value we like. So if we write $G_\mu(1)$, it does not infer that the random variable $\mathtt{X} = 1$. So in particular, $G_\mu(1) \neq \Pr(\mathtt{X} = 1)$. The values of the random variable $\mathtt{X}$ and the probability associated with that value correspond to the power of X and a coefficient. So if $\mu(j)X^j$ is a term of the PGF, it means that $\Pr(\mathtt{X} = j) = \mu(j)$.

There are also PGFs for the multivariate case [**?**]. If we have k discrete random variables and the probability mass function is given by

$$\mu\colon \mathbb{N}^k \to [0,1],\ (i_1, ..., i_k) \mapsto \Pr(\mathtt{X}_1 = i_1, ..., \mathtt{X}_k = i_k) \qquad [\mathbf{?}],$$

then the corresponding PGF is

$$G_\mu = \sum_{i_1,...,i_k \in \mathbb{N}} \mu(i_1, ..., i_k) X_1^{i_1} \cdots X_k^{i_k} \qquad [?].$$

We can operate on a PGF as on any other formal power series. So we can add and subtract PGFs, multiply them or obtain their derivation [?].

Computation of Stochastic Measurements

We can easily extract stochastic measurements such as the expected value or the variance from PGF of a univariate or multivariate discrete distribution. In addition, we can extract the sum of all the coefficients and single coefficients [?, ?]. Now, the different operations are explained for the uni- and multivariate case.

Univariate Case

Let `X` be a random variable and G_μ be the PGF for the univariate distribution of this variable.

- **Extraction of coefficients:** Extracting the coefficient a_0 is done by setting X to zero. Then, the PGF sums up to $\mu(0)$ since $X^i = 0^i = 0$ for all $i \neq 0$, so:

$$[X^0]f = \mu(0) = \sum_{i \in \mathbb{N}} \mu(i) 0^i = G_\mu(0)$$

  If we look at the first derivation we see that the coefficient of X^0 is $1 \cdot a_1$. Hence, if we set X to zero, the formal power series would sum up to $1 \cdot a_1$, so:

$$[X^1]f = \mu(1) = \sum_{i \in \mathbb{N}} i\mu(i) 0^{i-1} = G'_\mu(0)$$

  The first coefficient for which the extraction is a bit more complex is a_3. We look at the third derivation and see that the coefficient of X^0 is $3 \cdot 2 \cdot 1 \cdot a_3$. So we can extract the coefficient a_3 if we set X to zero and divided the value by $3 \cdot 2 \cdot 1 = 3! = 6$, we get:

$$[X^3]f = \mu(3) = \frac{1}{3!} \sum_{i \in \mathbb{N}} i(i-1)(i-2)\mu(i) 0^{i-3} = \frac{G'_\mu(0)}{3!}$$

We can generalize that and get the following form [**?**]:

$$[X^i]f(X) = \mu(i) = \frac{\partial^i f}{i!\partial X^i}(0)$$

We need to divide the derivation by $i!$, because we have to take into account that due to the process of the derivation all the coefficients get multiplied with the former powers.

- **Compute the probability mass of a PGF:** If we want to know the probability mass a specific PGF represents in total, we have to compute the sum of the coefficients which we get if we set the value of X to 1 [**?**].

$$G_\mu(1) = \sum_{i \in \mathbb{N}} \mu(i)$$

Since this series and all of its partial sums are bounded by 1, this series always converges [**?**].

- **Expected value**
We can express the mean or expected value as follows:

$$\begin{aligned} \mathrm{E}(\mathtt{X}) &= \sum_{i \in \mathbb{N}} i\mu(i) \\ &= \lim_{X \to 1} \sum_{i \in \mathbb{N}} i\mu(i)X^{i-1} \\ &= G'_\mu(1) \qquad [\mathbf{?}] \end{aligned}$$

- **Variance:** To extract the variance we use the formula

$$\mathrm{Var}(\mathtt{X}) = \mathrm{E}(\mathtt{X}^2) - (\mathrm{E}(\mathtt{X}))^2 \qquad [\mathbf{?}].$$

We already know from our results from above that

$$(\mathrm{E}(\mathtt{X}))^2 = (G'_\mu(1))^2$$

Now, we have to see how we can relate $E(\mathtt{X}^2)$ to the PGF.

$$\begin{aligned} E(\mathtt{X}^2) &= \sum_{i \in \mathbb{N}} i^2\mu(i) \\ &= \sum_{i \in \mathbb{N}} (i^2 - i + i)\mu(i) \\ &= \sum_{i \in \mathbb{N}} (i(i-1) + i)\mu(i) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i\in\mathbb{N}} i(i-1)\mu(i) + \sum_{i\in\mathbb{N}} i\mu(i) \\
&= \lim_{X\to 1} \sum_{i\in\mathbb{N}} i(i-1)\mu(i)X^{i-2} + \lim_{X\to 1} \sum_{i\in\mathbb{N}} i\mu(i)X^{i-1} \\
&= G''_\mu(1) + G'_\mu(1) \qquad [?]
\end{aligned}
$$

Other stochastic measurements like the standard deviation can also be obtained using the PGF [?].

Multivariate Case

For the multivariate case we get analogue formulas, but now we take into account that we have more than one random variable.

Let $\mathtt{X}_1, ..., \mathtt{X}_k$ be random variables and G_μ be the PGF for the multivariate distribution of those variables. Then we get the following formulas for extracting a coefficient, the sum of the coefficients, expected value and the variance.

- **Extraction of coefficients:** Now, if we want to extract a coefficient we have to do it with respect to all the unknown variables (proof in the appendix, page **??**).

$$\left[X_1^{i_1} \cdots X_k^{i_k}\right] G_\mu = \mu\left(i_1, ..., i_k\right) = \frac{\partial^{i_1}}{i_1! \partial X_1^{i_1}} \left(\frac{\partial^{i_2}}{i_2! \partial X_2^{i_2}} \left(... \left(\frac{\partial^{i_k} G_\mu}{i_k! \partial X_k^{i_k}} \right) ... \right) \right) (0, ..., 0)$$

- **Coefficient sum:** The sum of the coefficients is computed as before:

$$G_\mu(1, ..., 1) = \sum_{i_1, ..., i_k \in \mathbb{N}} \mu\left(i_1, ..., i_k\right)$$

  As all partial sums of this series are bounded by 1 and all coefficients are positive, this series converges [?].

- **Expected value of one random variable:** This is done as for the univariate case, only that we now take into account that we have more than just one random variable. That way, we have to use the first partial derivation of the PGF with respect to the random variable whose expected value we want to extract [?].

$$\mathrm{E}(\mathtt{X}_j) = \frac{\partial G_\mu}{\partial X_j}(1, ..., 1) \tag{2.1}$$

- **Variance:** As for the expected value we now extract the variance with respect to

one random variable X_j. For the univariate case this was done with the formula:

$$\mathrm{Var}\left(\mathtt{X}_j\right) = \mathrm{E}\left(\mathtt{X}_j^2\right) - \left(\mathrm{E}\left(\mathtt{X}_j\right)\right)^2$$

From Equation (2.1) we can deduce

$$\left(\mathrm{E}\left(\mathtt{X}_j\right)\right)^2 = \left(\frac{\partial G_\mu}{\partial X_j}(1,...,1)\right)^2$$

We now have to relate $\mathrm{E}\left(\mathtt{X}_j^2\right)$ to the PGF.

$$\begin{aligned}
\mathrm{E}\left(\mathtt{X}_j^2\right) = & \sum_{i_1,...,i_k \in \mathbb{N}} i_j^2 \mu(i_1,...,i_k) \\
= & \sum_{i_1,...,i_k \in \mathbb{N}} (i_j^2 - i_j + i_j)\mu(i_1,...,i_k) \\
= & \sum_{i_1,...,i_k \in \mathbb{N}} (i_j(i_j-1) + i_j)\mu(i_1,...,i_k) \\
= & \sum_{i_1,...,i_k \in \mathbb{N}} i_j(i_j-1)\mu(i_1,...,i_k) + \sum_{i_1,...,i_k \in \mathbb{N}} i_j \mu(i_1,...,i_k) \\
= & \lim_{X_1,...,X_k \to 1} \sum_{i_1,...,i_k \in \mathbb{N}} i_j(i_j-1)\mu(i_1,...,i_k) X_1^{i_1} \cdots X_j^{i_j-2} \cdots X_k^{i_k} \\
& + \lim_{X_1,...,X_k \to 1} \sum_{i_1,...,i_k \in \mathbb{N}} i_j \mu(i_1,...,i_k) X_1^{i_1} \cdots X_j^{i_j-1} \cdots X_k^{i_k} \\
= & \frac{\partial^2 G_\mu}{\partial X_j^2}(1,...,1) + \frac{\partial G_\mu}{\partial X_j}(1,...,1)
\end{aligned}$$

Together, we obtain the following formula:

$$\mathrm{Var}\left(\mathtt{X}_j\right) = \frac{\partial^2 G_\mu}{\partial X_j^2}(1,...,1) + \frac{\partial G_\mu}{\partial X_j}(1,...,1) - \left(\frac{\partial G_\mu}{\partial X_j}(1,...,1)\right)^2$$

These formulas should give as a good insight on how easily we can extract stochastic measurements from the PGF.

Convergence

When we looked at convergence for the formal power series, we mentioned the geometric series

$$\sum_{i \in \mathbb{N}} Z^i = \frac{1}{1-Z}.$$

It converges for a value of $|Z| < 1$. Because from now on we look only at PGFs we know that since all the coefficients are positive and their sum is less equal to one, series with such a format converge. To illustrate this, we give an example:
Example 2.2.1 (Closed Form of a PGF with the Rule for Geometric Series).
Suppose we have the following probability mass function:

$$\mu\colon \mathbb{N} \to [0,1],\ i \mapsto \left(\frac{1}{2}\right)^{i+1}$$

So the probability that `X` has the value zero is 0.5, the probability that `X` has the value one is 0.25 and so on. Based on that mass function we have the PGF G_μ:

$$G_\mu = \sum_{i\in\mathbb{N}} \mu(i)X = \sum_{i\in\mathbb{N}} \left(\frac{1}{2}\right)^{i+1} X^i = \frac{1}{2}\sum_{i\in\mathbb{N}} \left(\frac{1}{2}X\right)^i$$

With the rule for geometric series we get:

$$G_\mu = \frac{1}{2}\sum_{i\in\mathbb{N}} \left(\frac{1}{2}X\right)^i = \frac{1}{2}\frac{1}{1-\frac{1}{2}X} = \frac{1}{2-X}$$

△

2.3 Theoretical Backround on Semantics

This subsection is about denotational semantics and the underlying theory of domains. The semantics of a program is its meaning. There are several ideas on how we can specify the semantics of a program. One of these ideas is *denotational semantics*. To define a denotational semantics we have to gain some knowledge in domain theory first [**?**]. In this section, we give an introduction to domain theory and after that we will introduce the concept of denotational semantics for a program.

Domain Theory

Domain theory is the study of partially ordered sets, known as domains or *partial orders*. A partial order is a set with a binary relation such that it is reflexive, antisymmetric and transitive.
Definition 2.3.1 (Partial Order [**?**])**.**
Suppose L is a set and $\sqsubseteq \subseteq L \times L$ is a binary relation. Then $(L, \sqsubseteq)$ is called a partial order if the following holds for all $a, b, c \in L$:

- $a \sqsubseteq a$ (reflexivity).
- $a \sqsubseteq b \wedge b \sqsubseteq a \Rightarrow a = b$ (antisymmetry).
- $a \sqsubseteq b \wedge b \sqsubseteq c \Rightarrow a \sqsubseteq c$ (transitivity). △

Example 2.3.2. An example of a partial order is $(\mathbb{N}, \leq)$ [**?**]: It is reflexive since all natural numbers are less than or equal to themselves, it is anti-symmetric since if we have two distinct natural numbers one is less than or equal to the other and not the other way around, and this order is transitive. △

We can extend this definition of a partial order to define a *complete partial order*. In this case, complete means that any ascending chain $d_1 \sqsubseteq d_2 \sqsubseteq ...$ with $d_i \in L$ has a supremum [**?**]. A supremum is the least upper bound of a set [**?**]. Ascending chains are called ω-chains and can be defined as follows:

Definition 2.3.3 (ω-Chain [**?**])**.**
Suppose $(L, \sqsubseteq)$ is a partial order. Then $D = \{d_i | i \in \mathbb{N}\} \subseteq L$ is an ω-chain if

$$d_i \sqsubseteq d_{i+1} \qquad \text{for all } i \in D.$$

△

With that in mind a complete partial order can be defined:

Definition 2.3.4 (Complete Partial Order [**?**])**.**
Suppose L is a set and $\sqsubseteq \subseteq L \times L$ is a binary relation. Then $(L, \sqsubseteq)$ is called a complete partial order if

- $(L, \sqsubseteq)$ is a partial order.
- All ω-chains $D \subseteq L$ have a supremum in L.

If there exists an element $\bot \in L$ such that $\bot \sqsubseteq a$ for all $a \in L$, we call that element the *bottom element*. △

Example 2.3.5. We can now look at two partial orders and check whether they are also complete partial orders.

- Any partial order $(L, \sqsubseteq)$ where L is finite is a complete partial order [**?**]. If L is finite, every ω-chain also contains finitely many distinct elements. All elements of an ω-chain are related and therefore there is one element which is the maximum of this ω-chain [**?**].
- $(\mathbb{N}, \leq)$ is a partial order but not a complete partial order [**?**]. We can define the ω-chain $\{i | i \in \mathbb{N}\}$. This ω-chain has no supremum since it contains all natural numbers, which are unbounded [**?**]. △

Denotational Semantics

It is important to understand and formalize the meaning of a program: its *semantics*. One approach to representing a program's semantics is to define a function over a domain for each instruction a program can have. So we have an initial value and different functions for different instructions: we have a function modelling an assignment, a function for if-else-constructs etc. The semantics arising from that approach are called *denotational semantics*. This approach is different from other approaches, like operational semantics which focus on the computational steps the program indicates [**?**].

To compute the values of variables etc. resulting from a program we apply a function which represents the program on an initial value. This function describes the semantics of the program. The function we apply to that initial value is the composition of the functions for the different program instructions. Hence, the functions of a denotational semantic are *compositional*. Defining such a function for while-loops and recursion is a bit more involved. One commonly-used idea is to represent loops and recursion as the least fixed points of certain functions [**?**]. In this bachelor thesis, we take a slightly different approach: In our semantics, the result of the application of the while-loop is defined as the supremum of an ω-chain.

Chapter 3

Probability Generating Function Semantics

We defined a set of rules based on PGFs to formalize the semantics of a probabilistic program. Before we introduce the rules of our semantics, we define a relation $\sqsubseteq$ which relates two multivariate PGFs and show that the set of all PGFs and $\sqsubseteq$ form a complete partial order.

3.1 Our Domain

First, we introduce a set and a binary relation and show that this is a complete partial order. Let L be the set of all multivariate formal power series such that the sum of the coefficients of all those formal power series is less than or equal to 1. Formally:

$$L := \left\{ a = \sum_{i_1,\ldots,i_k \in \mathbb{N}} a_{i_1,\ldots,i_k} X_1^{i_1} \cdots X_k^{i_k} \middle| a_{i_1,\ldots,i_k} \in \mathbb{R}, \sum_{i_1,\ldots,i_k \in \mathbb{N}} \left[X_1^{i_1} \cdots X_k^{i_k} \right] a \leq 1, \right\}$$

Since the sum of the coefficients of any formal power series $a \in L$ is less than or equal to one, we could define a probability mass function, taking any coefficient $a_{i_1,\ldots,i_k}$ as the probability that $(\mathtt{X}_1 = i_1, ..., \mathtt{X}_k = i_k)$. In addition, all PGFs are formal power series such that the sum of all coefficients is less than or equal to one. So, L must contain all PGFs.

Now, we define the relation $\sqsubseteq$ on L as such that one formal power series is related to another one if and only if every coefficient is less than or equal to the *corresponding* coefficient of the other formal power series.

Definition 3.1.1 (Order on L)**.**
For two formal power series – $a, b \in L$ – holds $a \sqsubseteq b$ if and only if

$$\forall i_1, ..., i_k \in \mathbb{N} : \left[X_1^{i_1} \cdots X_k^{i_k}\right] a \leq \left[X_1^{i_1} \cdots X_k^{i_k}\right] b$$

Analogously $\sqsubset$ can be defined on L:

$$a \sqsubset b \text{ iff } a \sqsubseteq b \wedge a \neq b.$$

△

We can prove that $(L, \sqsubseteq)$ is a complete partial order. To prove that we have to check whether the order is reflexive, antisymmetric, transitive and whether every ω-chain has a supremum.

Theorem 3.1.2. $(L, \sqsubseteq)$ *is a complete partial order.*

Proof. Let $a, b, c \in L$, arbitrary.

First, we show that $(L, \sqsubseteq)$ is a partial order. So we prove that $\sqsubseteq$ is reflexive, antisymmetric and transitive.

- Reflexivity: Let $a \in L$. Then

$$\begin{aligned}
& \forall i_1, ..., i_k \in \mathbb{N} : \left[X_1^{i_1} \cdots X_k^{i_k}\right] a = \left[X_1^{i_1} \cdots X_k^{i_k}\right] a \\
\Rightarrow \quad & \forall i_1, ..., i_k \in \mathbb{N} : \left[X_1^{i_1} \cdots X_k^{i_k}\right] a \leq \left[X_1^{i_1} \cdots X_k^{i_k}\right] a \\
\Leftrightarrow \quad & a \sqsubseteq a \qquad \text{(Definition 3.1.1)}
\end{aligned}$$

- Antisymmetry: $a \sqsubseteq b \wedge b \sqsubseteq a \Rightarrow a = b$

$$\begin{aligned}
& a \sqsubseteq b \wedge b \sqsubseteq a \\
\Leftrightarrow \quad & \forall i_1, ..., i_k \in \mathbb{N} : \left[X_1^{i_1} \cdots X_k^{i_k}\right] a \leq \left[X_1^{i_1} \cdots X_k^{i_k}\right] b \\
& \wedge \forall i_1, ..., i_k \in \mathbb{N} : \left[X_1^{i_1} \cdots X_k^{i_k}\right] b \leq \left[X_1^{i_1} \cdots X_k^{i_k}\right] a \qquad \text{(Definition 3.1.1)} \\
\Rightarrow \quad & \forall i_1, ..., i_k \in \mathbb{N} : \left[X_1^{i_1} \cdots X_k^{i_k}\right] a = \left[X_1^{i_1} \cdots X_k^{i_k}\right] b \\
\Leftrightarrow \quad & a = b
\end{aligned}$$

- Transitivity: $a \sqsubseteq b \wedge b \sqsubseteq c \Rightarrow a \sqsubseteq c$

$$\begin{aligned}
& a \sqsubseteq b \wedge b \sqsubseteq c \\
\Leftrightarrow \quad & \forall i_1, ..., i_k \in \mathbb{N} : \left[X_1^{i_1} \cdots X_k^{i_k}\right] a \leq \left[X_1^{i_1} \cdots X_k^{i_k}\right] b
\end{aligned}$$

$$\begin{aligned} & \wedge\ \forall i_1, ..., i_k \in \mathbb{N} : \left[X_1^{i_1} \cdots X_k^{i_k}\right] b \leq \left[X_1^{i_1} \cdots X_k^{i_k}\right] c && \text{(Definition 3.1.1)} \\ \Rightarrow \quad & \forall i_1, ..., i_k \in \mathbb{N} : \left[X_1^{i_1} \cdots X_k^{i_k}\right] a \leq \left[X_1^{i_1} \cdots X_k^{i_k}\right] c \\ \Leftrightarrow \quad & a \sqsubseteq c && \text{(Definition 3.1.1)} \end{aligned}$$

We have shown that $(L, \sqsubseteq)$ is a partial order. In order to show that $(L, \sqsubseteq)$ is a *complete* partial order we need to prove that any ω-chain $D \subseteq L$ has a supremum s and that $s \in L$ [**?**]. The supremum has the property that all its coefficients are greater-equal to the corresponding coefficients of all $d \in D$. To obtain such a formal power series we take as coefficients the supremum of the corresponding coefficients of all $d \in D$. We show that the formal power series we construct by this approach is in fact the supremum of D. After that, we prove that the coefficients of s sum up to a value less than or equal to 1 such that $s \in L$.

Let $D = \{d_k | k \in \mathbb{N}\} \subseteq L$ be any ω-chain, such that $d_k \sqsubseteq d_{k+1}$ for all $k \in \mathbb{N}$.

Proposition 1: D has a supremum. This supremum is given by

$$s = \sum_{i_1, ..., i_k \in \mathbb{N}} \bigsqcup \left\{ \left[X_1^{i_1} \cdots X_k^{i_k}\right] d \middle| d \in D \right\} X_1^{i_1} \cdots X_k^{i_k}$$

First, we show that s is the supremum of D.

$$s = \sqcup D$$

- s is an upper bound of D.

  So for all $d \in D$, the following must hold:

  $$\begin{aligned} & d \sqsubseteq \sum_{i_1, ..., i_k \in \mathbb{N}} \bigsqcup \left\{ \left[X_1^{i_1} \cdots X_k^{i_k}\right] d \middle| d \in D \right\} X_1^{i_1} \cdots X_k^{i_k} \\ \Leftrightarrow\ & \forall i_1, ..., i_k \in \mathbb{N} : \left[X_1^{i_1} \cdots X_k^{i_k}\right] d \leq \bigsqcup \left\{ \left[X_1^{i_1} \cdots X_k^{i_k}\right] d \middle| d \in D \right\} \quad \text{(Definition 3.1.1)} \end{aligned}$$

  This is true by definition of the supremum.

- There is no other upper bound m of D with $m \sqsubset s$.

  Suppose there is such an m. Then there must exist some $i_1, ..., i_k \in \mathbb{N}$ such that for all $d \in D$, the following holds

  $$\left[X_1^{i_1} \cdots X_k^{i_k}\right] d \leq \left[X_1^{i_1} \cdots X_k^{i_k}\right] m < \left[X_1^{i_1} \cdots X_k^{i_k}\right] s = \bigsqcup \left\{ \left[X_1^{i_1} \cdots X_k^{i_k}\right] d \middle| d \in D \right\}.$$

  Since $\bigsqcup \left\{ \left[X_1^{i_1} \cdots X_k^{i_k}\right] d \middle| d \in D \right\}$ is the *least* upper bound of $\left\{ \left[X_1^{i_1} \cdots X_k^{i_k}\right] d \right\}$ that

is a contradiction.

Now, we need to show that $s \in L$.

Proposition 2: $s' = \sum_{i_1,...,i_k \in \mathbb{N}} \left[X_1^{i_1} \cdots X_k^{i_k} \right] s \leq 1$ such that $s \in L$.

So, we need to prove that the coefficient sum of s, s', converges to a value less than or equal to one. We take steps to do so:

- Our series has k countervariables. Most theorems on series deal with series with only one countervariable. In our case we introduce a series with only one countervariable. This series is a rearrangement of s'.
- We show that if this new series converges, it converges to the same value as s'.
- The series with only one countervariable converges to a value less than or equal to one.

We first construct a series with only one countervariable such that this new series is just a rearrangement of the original s'.

The set $\mathbb{N}^k$ is countable [**?**]. That means there exists a bijection

$$g \colon \mathbb{N} \to \mathbb{N}^k, i \mapsto (i_1, ..., i_k) \,.$$

We define

$$(X_1 \cdots X_k)^{(i_1,...,i_k)} = X_1^{i_1} \cdots X_k^{i_k}.$$

Then we construct the series

$$\sum_{i \in \mathbb{N}} \left[(X_1 \cdots X_k)^{g(i)} \right] s.$$

Since g is bijective this series is a rearrangement of s'.

If a series converges absolutely to a certain value, all rearrangements of that series also converge to that value. Absolute convergence means that if we take the absolute values of the terms of the original series the series still converges [**?**]. Since all of the coefficients are positive for s' and the new series with only one countervariable, those series converge absolutely if they converge. So:

$$\sum_{i_1,...,i_k \in \mathbb{N}} \left[X_1^{i_1} \cdots X_k^{i_k} \right] s = \sum_{i \in \mathbb{N}} \left[(X_1 \cdots X_k)^{g(i)} \right] s$$

Now, we would like to show that

$$\sum_{i\in\mathbb{N}} \left[(X_1 \cdots X_k)^{g(i)}\right] s \leq 1$$

By definition, the limit of the series is the same as the limit of the sequence of its partial sums,

$$\sum_{i\in\mathbb{N}} \left[(X_1 \cdots X_k)^{g(i)}\right] s = \lim_{n\to\infty} \sum_{i=0}^{n} \left[(X_1 \cdots X_k)^{g(i)}\right] s \qquad [?].$$

In addition, for all $n \in \mathbb{N}$, the following holds

$$\sum_{i=0}^{n} \left[(X_1 \cdots X_k)^{g(i)}\right] s = \sum_{i=0}^{n} \bigsqcup \left\{ \left[(X_1 \cdots X_k)^{g(i)}\right] d \middle| d \in D \right\} \qquad \text{by definition of } s.$$

Since D is an ω-chain, the sequence $\left\{ \left[(X_1 \cdots X_k)^{g(i)}\right] d_l \right\}_{l\in\mathbb{N}}$ is monotonic. In addition, all coefficients of all $d \in D$ are bounded by one. So, the sequence $\left\{ \left[(X_1 \cdots X_k)^{g(i)}\right] d_l \right\}_{l\in\mathbb{N}}$ is monotonic and bounded. As such it converges to its supremum [**?**]. Therefore we have

$$\bigsqcup \left\{ \left[(X_1 \cdots X_k)^{g(i)}\right] d \middle| d \in D \right\} = \lim_{l\to\infty} \left[(X_1 \cdots X_k)^{g(i)}\right] d_l.$$

We can insert that into our equation:

$$\begin{aligned}
\sum_{i=0}^{n} \left[(X_1 \cdots X_k)^{g(i)}\right] s &= \sum_{i=0}^{n} \bigsqcup \left\{ \left[(X_1 \cdots X_k)^{g(i)}\right] d \middle| d \in D \right\} \\
&= \sum_{i=0}^{n} \lim_{l\to\infty} \left[(X_1 \cdots X_k)^{g(i)}\right] d_l \\
&= \lim_{l\to\infty} \sum_{i=0}^{n} \left[(X_1 \cdots X_k)^{g(i)}\right] d_l \\
&\qquad\qquad \text{(property of the limit of a sequence [?])}
\end{aligned}$$

Since all the coefficients are positive, for all $l \in \mathbb{N}$

$$\sum_{i=0}^{n} \left[(X_1 \cdots X_k)^{g(i)}\right] d_l \leq \sum_{i=0}^{\infty} \left[(X_1 \cdots X_k)^{g(i)}\right] d_l.$$

Because all coefficients are positive any rearrangement of the later series converges to the same value (if it converges). So we have

$$\sum_{i=0}^{\infty} \left[(X_1 \cdots X_k)^{g(i)}\right] d_l = \sum_{i_1,\ldots,i_k\in\mathbb{N}} \left[X_1^{i_1} \cdots X_k^{i_k}\right] d_l.$$

$\sum_{i_1,\ldots,i_k\in\mathbb{N}} \left[X_1^{i_1}\cdots X_k^{i_k}\right] d_l$ is bounded by 1 and as such converges to a value less than or equal to one. With the estimation of the elements of the two sequences:

$$\begin{aligned}\lim_{l\to\infty}\sum_{i=0}^{n}\left[(X_1\cdots X_k)^{g(i)}\right] d_l &\leq \lim_{l\to\infty}\sum_{i=0}^{\infty}\left[(X_1\cdots X_k)^{g(i)}\right] d_l \\ &= \lim_{l\to\infty}\sum_{i_1,\ldots,i_k\in\mathbb{N}}\left[X_1^{i_1}\cdots X_k^{i_k}\right] d_l \\ &\leq 1.\end{aligned} \qquad [?]$$

We use that for our estimation:

$$\begin{aligned}\sum_{i=0}^{n}\left[(X_1\cdots X_k)^{g(i)}\right] s &= \lim_{l\to\infty}\sum_{i=0}^{n}\left[(X_1\cdots X_k)^{g(i)}\right] d_l \\ &\leq \lim_{l\to\infty}\sum_{i_1,\ldots,i_k\in\mathbb{N}}\left[X_1^{i_1}\cdots X_k^{i_k}\right] d_l \\ &\leq 1\end{aligned}$$

Since the sequence of the partial sums, $\left\{\sum_{i=0}^{n}\left[(X_1\cdots X_k)^{g(i)}\right] s\right\}_{n\in\mathbb{N}}$ is monotonic (all coefficients are positive) and bounded by one, the limit of the sequence of partial sums is bounded by 1 [?].

$$\sum_{i_1,\ldots,i_k\in\mathbb{N}}\left[X_1^{i_1}\cdots X_k^{i_k}\right] s = \sum_{i\in\mathbb{N}}\left[(X_1\cdots X_k)^{g(i)}\right] s = \lim_{n\to\infty}\sum_{i=0}^{n}\left[(X_1\cdots X_k)^{g(i)}\right] s \leq 1$$

□

In fact, the supremum of the sums of coefficients of $d \in D$ equals the sum of coefficients of s. This proposition can be useful in other contexts.

Lemma 3.1.3. *For all ω-chains $D \subseteq L$:*

$$\bigsqcup\left\{\sum_{i_1,\ldots,i_k\in\mathbb{N}}\left[X_1^{i_1}\cdots X_k^{i_k}\right] d \middle| d\in D\right\} = \sum_{i_1,\ldots,i_k\in\mathbb{N}}\bigsqcup\left\{\left[X_1^{i_1}\cdots X_k^{i_k}\right] d \middle| d\in D\right\}.$$

The proof can be found in the appendix (page **??**).

In the following we will need the definition of convergence for the set L. The definition of convergence requires a metric on the set. Therefore, we define a metric on L and prove that it is one. A metric is a mapping d from $L\times L$ to $\mathbb{R}$ which is 0 if and only if the input values are the same, always positive, symmetric and satisfies the triangle inequality [?]. The mapping we define in the following and for which we will prove that it is a metric

on L maps two values of L to the sum of an absolute values of the differences of their coefficients.

Definition 3.1.4 (Metric on L)**.**

$$d\colon L \times L \to \mathbb{R}, (a,b) \mapsto \sum_{i_1,\ldots,i_k \in \mathbb{N}} \left| \left[X_1^{i_1} \cdots X_k^{i_k} \right] a - \left[X_1^{i_1} \cdots X_k^{i_k} \right] b \right|$$

△

Theorem 3.1.5. *(L,d) is a metric space.*

The proof can be found in the appendix (page **??**).

Since there is a general definition of convergence in metric spaces, we can introduce that now with regard to our metric space. The definition of convergence is the same as for $\mathbb{R}$ but with our metric d instead of the absolute value.

Definition 3.1.6 (Convergence in (L,d) [**?**])**.**
A sequence $\{a_n\}_{n\in\mathbb{N}}$ in (L,d) converges to $a \in L$ if for all $\epsilon > 0$ there exists n_ϵ such that for all $n > n_\epsilon$ there is $d(a_n, n) < \epsilon$.

As for sequences in $\mathbb{R}$ for sequences in L holds that if they are monotonic and bounded, they converge towards their supremum. Since all monotonic sequences in L have a supremum, all monotonic sequences in L are bounded.

Lemma 3.1.7. *All monotonic sequences in L converge. The limit of a sequence is its supremum.*

The proof can be found in the appendix (page **??**).

3.2 Probability Generating Function Rules

Since we have shown that $(L, \sqsubseteq)$ is a complete partial order, we can now define rules for describing the semantics of a program. Those rules were defined by the research group. First of all, we describe informally the effect of the different statements. Our programs consist of the following constructs:

- `skip`: We do not do anything and just skip the line.
- `abort`: We immediately do not terminate. A program which uses abort will never terminate successfully, either because it encounters an error or because it runs forever.
- $x_j := e$: The value of the program variable x_j is set to the evaluation of the

expression e. In addition, any variable of the program can only be set to a natural number.

- $\mathtt{P}_1;\mathtt{P}_2$: We have a sequence of two programs. $\mathtt{P}_1$ is executed first and after that $\mathtt{P}_2$ is executed.
- $\{\mathtt{P}_1\}[p]\{\mathtt{P}_2\}$: We have a probabilistic choice between the programs $\mathtt{P}_1$ and $\mathtt{P}_2$. With probability p we continue our execution with program $\mathtt{P}_1$or switch to program $\mathtt{P}_2$ with probability $(1-p)$.
- `if` (B) $\{\mathtt{P}_1\}$ `else` $\{\mathtt{P}_2\}$: We have a conditional choice based on the boolean condition B. If the condition is satisfied, we will execute $\mathtt{P}_1$; if not, we will execute $\mathtt{P}_2$.
- `while` $(B)\{\mathtt{P}\}$: We have a loop with a boolean condition B. As long as that condition is satisfied we execute the loop body $\mathtt{P}$.

At each point of the program each program variable has a certain value with a certain probability. So, our program states are distributions over variables. We represent those probability distributions as PGFs. All PGFs G_μ are in L as L is the set of all PGFs. Therefore, the probability mass of the distribution corresponds to the probability of termination. Initially, the probability of termination is 1 and all variables are set to zero. So, our initial value is $1 = 1X_1^0 \cdots X_k^0$ if variables $x_1, ..., x_k$ occur in the program. We can now think about functions corresponding to the different programming constructs. Those functions modify the distributions of the variables. We denote the function corresponding to a program construct with brackets $[\![]\!]$. The functions get a PGF as input and modify this such that the modified PGF represents the effect the instruction had on the program state.

- $[\![\mathtt{skip}]\!](G_\mu) = (G_\mu)$: The input PGF is not modified, because `skip` does not do anything.
- $[\![\mathtt{abort}]\!](G_\mu) = 0$: A program which uses `abort` will never terminate successfully. Hence, the probability mass of our distribution and the probabilities of the variables having any value are zero afterwards.
- $[\![x_j := e]\!](G_\mu) = \sum_{i_1,...,i_k \in \mathbb{N}} \mu(i_1, ..., i_k) X_1^{i_1} \cdots X_j^{e(i_1,...,i_k)} \cdots X_k^{i_k}$: In a PGF, if the unknown variables $X_1, ..., X_k$ have the exponents $i_1, ..., i_k$, the probability that the program state is $(x_1 = i_1, ..., x_k = i_k)$ equals the value of the corresponding coefficient $\mu(i_1, ..., i_k)$. So, the value of an exponent relates to the value of the corresponding program variable. Therefore, we have to modify the exponent of the unknown variable X_j according to e.

- $[\![\mathtt{P}_1;\mathtt{P}_2]\!]\,(G_\mu) = [\![\mathtt{P}_2]\!]\,([\![\mathtt{P}_1]\!]\,(G_\mu))$: We modify the given PGF according to $[\![\mathtt{P}_1]\!]$ and $[\![\mathtt{P}_2]\!]$. First, we have to apply $[\![\mathtt{P}_1]\!]$ and after that we apply $[\![\mathtt{P}_2]\!]$ to the PGF we get by applying $[\![\mathtt{P}_1]\!]$ on the given PGF.

- $[\![\{P_1\}\,[p]\,\{P_2\}]\!]\,(G_\mu) = p\cdot[\![\mathtt{P}_1]\!]\,(G_\mu)+(1-p)\cdot[\![\mathtt{P}_2]\!]\,(G_\mu)$: We have to apply $[\![\mathtt{P}_1]\!]$ to the input PGF and multiply the resulting PGF by p in order to scale the probabilities according to probabilistic decision made. We do the same thing with the input PGF, $[\![\mathtt{P}_2]\!]$ and $(1-p)$. We sum the two resulting PGFs.

- $\langle G_\mu\rangle|_B$: We restrict a PGF to the boolean condition B. That means we only look at the parts of the PGF where the exponents of the unknown variables satisfy the boolean condition B. The formal definition is introduced later.

- $[\![\mathtt{if\ (B)\ \{P_1\}\ else\ \{P_2\}}]\!]\,(G_\mu) = [\![\mathtt{P}_1]\!]\,(\langle G_\mu\rangle|_B) + [\![\mathtt{P}_2]\!]\,(\langle G_\mu\rangle|_{\neg B})$: We have to restrict the PGF we get to the parts which satisfy B. $[\![\mathtt{P}_1]\!]$ is applied to this part. $[\![\mathtt{P}_2]\!]$ is applied to the part which does not satisfy B.

So far, we left out $[\![\mathtt{while}(B)\{\mathtt{P}\}]\!]$ because it is more complicated than the other rules. We will consider loops later.

Definition 3.2.1 (Rules for PGF Semantics)**.**
Let G_μ be any PGF. The definitions of the rules of our semantics are given by:

$$
\begin{aligned}
[\![\mathtt{skip}]\!]\,(G_\mu) &= G_\mu\\
[\![\mathtt{abort}]\!]\,(G_\mu) &= 0\\
[\![x_j := e]\!]\,(G_\mu) &= \sum_{i_1,\ldots,i_k\in\mathbb{N}} \mu(i_1,\ldots,i_k)X_1^{i_1}\cdots X_j^{e(i_1,\ldots,i_k)}\cdots X_k^{i_k}\\
[\![\mathtt{P}_1;\mathtt{P}_2]\!]\,(G_\mu) &= [\![\mathtt{P}_2]\!]\,([\![\mathtt{P}_1]\!]\,(G_\mu))\\
[\![\{P_1\}\,[p]\,\{P_2\}]\!]\,(G_\mu) &= p\cdot[\![\mathtt{P}_1]\!]\,(G_\mu) + (1-p)\cdot[\![\mathtt{P}_2]\!]\,(G_\mu)\\
[\![\mathtt{if\ (B)\ \{P_1\}\ else\ \{P_2\}}]\!]\,(G_\mu) &= [\![\mathtt{P}_1]\!]\,(\langle G_\mu\rangle|_B) + [\![\mathtt{P}_2]\!]\,(\langle G_\mu\rangle|_{\neg B})
\end{aligned}
$$

△

To define the rules properly we need to define the bottom-element 0 and restriction. Those are defined as follows:

Definition 3.2.2 (0 in PGF Semantics)**.**
0 is the bottom element of $(L, \sqsubseteq)$.

$$0 = \sum_{i_1,...,i_k \in \mathbb{N}} \mu(i_1, ..., i_k) X_1^{i_1} \cdots X_k^{i_k} \text{ with } \mu : \mathbb{N}^k \to [0,1], \vec{x} \mapsto 0$$

△

Restriction is defined as follows:
Definition 3.2.3 (Restriction)**.**
Let G_μ be any PGF.

$$\langle G_\mu \rangle|_B = \sum_{i_1,...,i_k \in \mathbb{N}} \begin{cases} \mu(i_1, ..., i_k) X_1^{i_1} \cdots X_k^{i_k} & i_1, ..., i_k \models B \\ 0 & \text{otherwise} \end{cases}$$

△

Example 3.2.4. To illustrate how these rules work we give an example of a program which contains some of the constructs for which we already have defined rules.

Program 1

```
1: {c:=0} [0.5] {c:=1};
2: if (c≠0) {
3:       {c:=0} [0.5] {c:=1}
4: } else {
5:       abort
6: }
```

With the notation $\mathtt{P}_{i,j}$ we refer to program from line i to line j of the Program 1.

Now, we want to apply our rules and get the probability distribution the program has after the execution. The initial PGF is $G_\mu = 1 \cdot C^0 = 1$.

$$\begin{aligned}
&[\![\mathtt{P}_{1,5}]\!](1) \\
&= [\![\{c := 0\}[0.5]\{c := 1\}; \mathtt{P}_{2,5}]\!](1) \\
&\qquad \text{sequential rule } [\![\mathtt{P}_1; \mathtt{P}_2]\!](G_\mu) = [\![\mathtt{P}_2]\!]([\![\mathtt{P}_1]\!](G_\mu)) \\
&= [\![\mathtt{P}_{2,5}]\!]([\![\{c := 0\}[0.5]\{c := 1\}]\!](1)) \\
&\qquad \text{probabilistic choice rule } [\![\{P_1\}\,[p]\,\{P_2\}]\!](G_\mu) = p \cdot [\![\mathtt{P}_1]\!](G_\mu) + (1-p) \cdot [\![\mathtt{P}_2]\!](G_\mu) \\
&= [\![\mathtt{P}_{2,5}]\!](0.5 \cdot ([\![c := 0]\!](1)) + 0.5 \cdot ([\![c := 1]\!](1))) \\
&\qquad \text{assignment rule } [\![x_j := e]\!](G_\mu) = \sum_{i_1,...,i_k \in \mathbb{N}} \mu(i_1, ..., i_k) X_1^{i_1} \cdots X_j^{e(i_1,...,i_k)} \cdots X_k^{i_k} \\
&= [\![\mathtt{P}_{2,5}]\!]\left(0.5 \cdot C^0 + 0.5 \cdot C^1\right)
\end{aligned}$$

definition of $[\![\mathtt{P}_{2,5}]\!]$

$$= [\![\mathtt{if}(c \neq 0)\mathtt{then}\,\{\mathtt{P}_{3,3}\}\,\mathtt{else}\,\{\mathtt{P}_{5,5}\}]\!]\left(0.5 \cdot C^0 + 0.5 \cdot C^1\right)$$

if-else rule $[\![\mathtt{if\ (B)\ \{P_1\}\ else\ \{P_2\}}]\!](G_\mu) = [\![\mathtt{P}_1]\!]\left(\langle G_\mu \rangle|_B\right) + [\![\mathtt{P}_2]\!](G_\mu \neg B)$

$$= [\![\mathtt{P}_{3,3}]\!]\left(\left\langle 0.5 \cdot C^0 + 0.5 \cdot C^1 \right\rangle\Big|_{c\neq 0}\right) + [\![\mathtt{P}_{5,5}]\!]\left(\left\langle 0.5 \cdot C^0 + 0.5 \cdot C^1 \right\rangle\Big|_{c=0}\right)$$

restriction $\langle G_\mu \rangle|_B = \sum_{i_1,\ldots,i_k \in \mathbb{N}} \begin{cases} \mu(i_1,\ldots,i_k) X_1^{i_1} \cdots X_k^{i_k} & i_1,\ldots,i_k \models B \\ 0 & \text{otherwise} \end{cases}$

$$= [\![\mathtt{P}_{3,3}]\!]\left(0.5 \cdot C^1\right) + [\![\mathtt{P}_{5,5}]\!]\left(0.5 \cdot C^0\right)$$

definition of $[\![\mathtt{P}_{3,3}]\!]$ and $[\![\mathtt{P}_{5,5}]\!]$

$$= [\![\{c := 0\}[0.5]\{c := 1\}]\!]\left(0.5 \cdot X^1 C^1\right) + [\![\mathtt{abort}]\!]\left(0.5 \cdot C^0\right)$$

probabilistic choice rule, abort rule $[\![\mathtt{abort}]\!](G_\mu) = 0$

$$= 0.5 \cdot \left([\![c := 0]\!]\left(0.5 C^1\right)\right) + 0.5 \cdot \left([\![c := 1]\!]\left(0.5 C^1\right)\right) + 0$$

assignment rule

$$= 0.25 \cdot C^0 + 0.25 \cdot C^1$$

△

In the chapter about PGFs, we saw that we can extract several pieces of information from a PGF. Among others, we can see that at the end ($c = 0$) holds with probability 0.25 and ($c = 1$) with probability 0.25. In addition, we can analyse the probability mass of this PGF. The probability mass of the distribution of the program variables is 0.5. That tells us that Program 1 terminates with probability $1 - 0.5 = 0.5$.

3.3 While-Loops in Probability Generating Function Semantics

In order to understand in what way a function $[\![\mathtt{while}(B)\{\mathtt{P}\}]\!]$ modifies a given PGF, we can take steps and describe first the scenario of a certain number of executions of a while-loop, $\mathtt{while}^{\leq k}(B)\{\mathtt{P}\}$. If we execute the while-loop zero-times, the input PGF is not modified. If we execute the while-loop $(k+1)$-times, we apply $\mathtt{P}$ on the input PGF restricted to B and have to apply the while-loop k times on the resulting PGF.

This can be formulated as an if-else-construct for which we already defined a rule.

Definition 3.3.1 (Loop-Unrolling)**.**
For all $k \in \mathbb{N}$, the loop-unrolling is defined as follows:

$$\begin{aligned}\left[\!\left[\texttt{while}^{\leq 0}(B)\,\{\texttt{P}\}\right]\!\right](G_\mu) &= G_\mu \\ \left[\!\left[\texttt{while}^{\leq k+1}(B)\,\{\texttt{P}\}\right]\!\right](G_\mu) &= \left[\!\left[\texttt{if } (B)\left\{\texttt{P};\texttt{while}^{\leq k}(B)\,\{\texttt{P}\}\right\} \texttt{ else } \{\texttt{skip}\}\right]\!\right](G_\mu)\end{aligned}$$

△

Example 3.3.2. We see how the loop-unrolling works if we look at the following example.

Program 2

```
1: x:=0;
2: {c:=0} [0.5] {c:=1}
3: while (c≠0) {
4:         x:=x+1;
5:         {c:=0} [0.5] {c:=1}
6: }
```

The PGF resulting from the first part of the program is obtained analogously to Program 1. Therefore $[\![\texttt{P}_{1,2}]\!](1X^0C^0) = 0.5X^0C^0 + 0.5X^0C^1$. To illustrate how the loop-unrolling works we do one loop-unrolling.

$$\begin{aligned}&\left[\!\left[\texttt{while}^{\leq 1}(B)\,\{\texttt{P}\}\right]\!\right]\left(0.5X^0C^0 + 0.5X^0C^1\right) \\ &\qquad \text{definition of loop-unrolling for } k=1 \\ &\qquad \left[\!\left[\texttt{while}^{\leq 1}(B)\,\{\texttt{P}\}\right]\!\right](G_\mu) = \left[\!\left[\texttt{if } (B)\left\{\texttt{P};\texttt{while}^{\leq 0}(B)\,\{\texttt{P}\}\right\} \texttt{ else } \{\texttt{skip}\}\right]\!\right](G_\mu) \\ &= \left[\!\left[\texttt{if } (B)\left\{\texttt{P};\texttt{while}^{\leq 0}(B)\,\{\texttt{P}\}\right\} \texttt{ else } \{\texttt{skip}\}\right]\!\right]\left(0.5X^0C^0 + 0.5X^0C^1\right) \\ &\qquad \text{if-else rule} \\ &= \left[\!\left[\texttt{P};\texttt{while}^{\leq 0}(B)\,\{\texttt{P}\}\right]\!\right]\left(\left\langle 0.5X^0C^0 + 0.5X^0C^1\right\rangle\Big|_{c\neq 0}\right) + \left[\!\left[\texttt{skip}\right]\!\right]\left(\left\langle 0.5X^0C^0 + 0.5X^0C^1\right\rangle\Big|_{c=0}\right) \\ &\qquad \text{definition of restriction} \\ &= \left[\!\left[\texttt{P};\texttt{while}^{\leq 0}(B)\,\{\texttt{P}\}\right]\!\right]\left(0.5X^0C^1\right) + \left[\!\left[\texttt{skip}\right]\!\right]\left(0.5X^0C^0\right) \\ &\qquad \text{sequence rule and skip rule } \left[\!\left[\texttt{skip}\right]\!\right](G_\mu) = G_\mu \\ &= \left[\!\left[\texttt{while}^{\leq 0}(B)\,\{\texttt{P}\}\right]\!\right]\left(\left[\!\left[\texttt{P}\right]\!\right]\left(0.5X^0C^1\right)\right) + 0.5X^0C^0 \\ &\qquad \text{definition of P} \\ &= \left[\!\left[\texttt{while}^{\leq 0}(B)\,\{\texttt{P}\}\right]\!\right]\left(\left[\!\left[x := x+1; \{c:=0\}[0.5]\{c:=1\}\right]\!\right]\left(0.5X^0C^1\right)\right) + 0.5X^0C^0 \\ &\qquad \text{sequence rule and assignment rule}\end{aligned}$$

$$= \left[\!\left[\texttt{while}^{\leq 0}(B)\,\{\texttt{P}\}\right]\!\right]\left(\left[\!\left[\{c := 0\}[0.5]\{c := 1\}\right]\!\right]\left(0.5X^1C^1\right)\right) + 0.5X^0C^0$$

rule for probabilistic choice

$$= \left[\!\left[\texttt{while}^{\leq 0}(B)\,\{\texttt{P}\}\right]\!\right]\left(0.25X^1C^0 + 0.25X^1C^1\right) + 0.5X^0C^0$$

loop-unrolling for $k = 0$ $\left[\!\left[\texttt{while}^{\leq 0}(B)\,\{\texttt{P}\}\right]\!\right](G_\mu) = G_\mu$

$$=0.25X^1C^0 + 0.25X^1C^1 + 0.5X^0C^0$$

△

We see that the loop-unrolling gives only an idea on what happens when executing the loop. For the first and the third term, the coefficients will only increase with further loop-unrollings since those terms are not part of the PGF restricted to $B := (c \neq 0)$. The coefficient of the second term will be further modified by additional executions of the loop since here $c = 1$ and the condition B is satisfied.

Our approach to model a while rule is to define a function which equals k-times applied on the input PGF the k-th loop-unrolling.

Definition 3.3.3 (Function for Describing While-Loops)**.**

$$H_{\texttt{P},B} : L \to L,\ K \mapsto \left[\!\left[\texttt{P}\right]\!\right](\langle K\rangle|_B) + \langle K\rangle|_{\neg B}\,, \qquad \text{for all while-loops } \texttt{while}(B)\{\texttt{P}\}.$$

As mentioned before, we constructed $H_{\texttt{P},B}$ such that the following relation holds:

Lemma 3.3.4. *For all $k \in \mathbb{N}$, the following holds:*

$$H^k_{\texttt{P},B}\,(G_\mu) = \left[\!\left[\texttt{while}^{\leq k}(B)\,\{\texttt{P}\}\right]\!\right](G_\mu)\,, \qquad \text{for } H_{\texttt{P},B} \text{ from Definition 3.3.3.}$$

The proof is done by induction over $k \in \mathbb{N}$ and can be found in the appendix on page **??**.

If we restrict the PGF resulting from this function to $\neg B$ we get a part of the PGF where no coefficient will decrease with further loop-unrollings. So the PGF resulting from an execution of a while-loop is the supremum of all PGFs we get by loop-unrolling and restricting that to $\neg B$.

Definition 3.3.5 (While Rule)**.**

For all PGFs G_μ, $\left[\!\left[\texttt{while}(B)\{\texttt{P}\}\right]\!\right](G_\mu)$ is defined as follows

$$\left[\!\left[\texttt{while}(B)\{\texttt{P}\}\right]\!\right](G_\mu) = \bigsqcup\left\{\left\langle H^k_{\texttt{P},B}\,(G_\mu)\right\rangle\Big|_{\neg B}\Big| k \in \mathbb{N}\right\}, \quad \text{for } H_{\texttt{P},B} \text{ from Definition 3.3.3.}$$

△

We can prove that $\left\{\left\langle H^k_{\texttt{P},B}\,(G_\mu)\right\rangle\Big|_{\neg B}\Big| k \in \mathbb{N}\right\}$ is an ω-chain. For that we have to show that it is a subset of L and that it is monotonic. Due to the fact the $(L, \sqsubseteq)$ is a complete

partial order the supremum of this ω-chain always exists and is in L.

Lemma 3.3.6.

$$\left\{ \left\langle H^k_{P,B}\left(G_\mu\right)\right\rangle\Big|_{\neg B} \Big| k \in \mathbb{N} \right\} \subseteq L$$

This proof is part of the proof that the image of any rule is a subset of L. This is proved by induction over the structure of P. The proof that the image of any rule is a subset of L can be found in the appendix (page **??**).

The next observation was that $\left\{ \left\langle H^k_{\mathtt{P},B}\left(G_\mu\right)\right\rangle\Big|_{\neg B} \Big| k \in \mathbb{N} \right\}$ is an ω-chain. Any element of the set contains only terms whose values can only increase with further loop-unrollings.

Lemma 3.3.7. $\left\{ \left\langle H^k_{P,B}\left(G_\mu\right)\right\rangle\Big|_{\neg B} \Big| k \in \mathbb{N} \right\}$ *is an ω-chain, so for all $k \in \mathbb{N}$, the following holds:* $\left\langle H^k_{P,B}\left(G_\mu\right)\right\rangle\Big|_{\neg B} \sqsubseteq \left\langle H^{k+1}_{P,B}\left(G_\mu\right)\right\rangle\Big|_{\neg B}$

The proof uses the fact that restriction is linear. It can be found in the appendix (page **??**).

As $(L, \sqsubseteq)$ is a complete partial order, the sought-after supremum always exists.

Corollary 3.3.8. $\bigsqcup \left\{ \left\langle H^k_{P,B}\left(G_\mu\right)\right\rangle\Big|_{\neg B} \Big| k \in \mathbb{N} \right\}$ *always exists.*

Example 3.3.9. We will illustrate this definition of the while-rule for the while-loop of Program 2.

```
1: while (c≠0) {
2:         x:=x+1;
3:        {c:=0} [0.5] {c:=1}
4: }
```

The PGF before we enter the while-loop is

$$G_\mu = 0.5X^0C^0 + 0.5X^0C^1.$$

In order to obtain the supremum of $\left\{ \left\langle H^k_{\mathtt{P},B}\left(G_\mu\right)\right\rangle\Big|_{\neg B} \Big| k \in \mathbb{N} \right\}$ we try to find a closed form for $\left\langle H^k_{\mathtt{P},B}\left(G_\mu\right)\right\rangle\Big|_{c=0}$. First, we prove that there is a closed form of $H^k_{\mathtt{P},B}\left(G_\mu\right)$ for all $k \in \mathbb{N}$.

For all $k \in \mathbb{N}$, the following holds

$$H^k_{\mathtt{P},B}\left(G_\mu\right) = 0.5X^0C^0 \cdot \sum_{i=0}^{k} \left((0.5X)^i \right) + 0.5^{k+1}C^1X^k. \tag{3.1}$$

The proof is done by induction over $k \in \mathbb{N}$ and can be found in the appendix (page **??**).

Now, we can find a closed form for $\left\langle H^k_{\mathtt{P},B}\left(G_\mu\right)\right\rangle\Big|_{c=0}$. For all $k \in \mathbb{N}$, the following holds

$$\begin{aligned} H^k_{\mathtt{P},B}\left(G_\mu\right)\neg B &= \left\langle 0.5X^0C^0 \cdot \sum_{i=0}^{k}\left((0.5X)^i\right) + 0.5^{k+1}C^1X^k \right\rangle\Big|_{\neg B} \quad \text{(with Equation (3.1))} \\ &= 0.5X^0C^0 \cdot \sum_{i=0}^{k}\left((0.5X)^i\right) \end{aligned}$$

With that in mind we have:

$$\bigsqcup\left\{\left\langle H^k_{\mathtt{P},B}\left(G_\mu\right)\right\rangle\Big|_{\neg B}\Big| k \in \mathbb{N}\right\} = \bigsqcup\left\{0.5X^0C^0 \cdot \sum_{i=0}^{k}\left((0.5X)^i\right)\Big| k \in \mathbb{N}\right\}$$

We can define a sequence

$$\left\{0.5X^0C^0 \cdot \sum_{i=0}^{k}\left((0.5X)^i\right)\right\}_{k\in\mathbb{N}}.$$

This sequence is monotonic since the set is an ω-chain (Lemma 3.3.7). So it converges towards its supremum (Lemma 3.1.7). Since the sequence converges, the limit is $0.5X^0C^0 \cdot \sum\limits_{i=0}^{\infty}\left((0.5X)^i\right)$. Therefore, we have:

$$\begin{aligned} \bigsqcup\left\{0.5X^0C^0 \cdot \sum_{i=0}^{k}\left((0.5X)^i\right)|k \in \mathbb{N}\right\} &= 0.5X^0C^0 \cdot \sum_{i=0}^{\infty}\left((0.5X)^i\right) \\ &= 0.5 \cdot \sum_{i=0}^{\infty}\left((0.5X)^i\right). \end{aligned}$$

In the first chapter, we have seen the geometric series. So we have:

$$\begin{aligned} \bigsqcup\left\{0.5X^0C^0 \cdot \sum_{i=0}^{k}(0.5X)^i|k \in \mathbb{N}\right\} &= 0.5 \cdot \sum_{i=0}^{\infty}(0.5X)^i \\ &= \frac{0.5}{1-0.5X} \\ &= \frac{1}{2-X} \end{aligned}$$

Taking everything into consideration,

$$\begin{aligned} [\![\texttt{while}(B)\{\mathtt{P}\}]\!]\left(G_\mu\right) &= \bigsqcup\left\{\left\langle H^k_{\mathtt{P},B}\left(G_\mu\right)\right\rangle\Big|_{\neg B}\Big| k \in \mathbb{N}\right\} \\ &= \bigsqcup\left\{0.5X^0C^0 \cdot \sum_{i=0}^{k}(0.5X)^i|k \in \mathbb{N}\right\} \end{aligned}$$

$$= \frac{1}{2 - X}$$

△

3.4 Properties of Probability Generating Function Rules

Now, that we have defined rules for all programming constructs, we can look at properties of $[\![\mathtt{P}]\!]$. We check whether $[\![\mathtt{P}]\!](G_\mu)$ is always well-defined and linear. In fact, both of these propositions hold.

Theorem 3.4.1. *$[\![\mathtt{P}]\!](G_\mu)$ is always well-defined. This means that $[\![\mathtt{P}]\!](G_\mu)$ is always a PGF such that $[\![\mathtt{P}]\!](G_\mu) \in L$ for all PGFs G_μ.*

The proof consists of two steps: First we show by induction over the structure of P that the probability mass of the PGF $[\![\mathtt{P}]\!](G_\mu)$ is less than or equal to the probability mass of G_μ. With that we can deduce well-definedness of $[\![\mathtt{P}]\!](G_\mu)$. The proof can be found in the appendix (page **??**).

Theorem 3.4.2. *$[\![\mathtt{P}]\!](G_\mu)$ is linear. That means for all $a \in [0,1]$ and all PGFs G_{μ_1}, G_{μ_2}:*

$$[\![\mathtt{P}]\!](a \cdot G_{\mu_1} + G_{\mu_2}) = a \cdot [\![\mathtt{P}]\!](G_{\mu_1}) + [\![\mathtt{P}]\!](G_{\mu_2}) \qquad [?]$$

The proof is done by induction over the structure of P and can be found in the appendix (page **??**).

Chapter 4

Proof Rules and Their Application

We have seen that it is quite complicated to obtain the PGF resulting from a while-loop. In order to simplify this, we developed some proof rules. If we know that the body of a while-loop has a certain form, we can then directly deduce the resulting PGF. In the following, we introduce some proof rules and present one form of a while-loop those rules can always be applied to.

4.1 Proof Rules

To get the PGF resulting from a while-loop we always have to obtain $\bigsqcup \left\{ \left\langle H^k_{\mathtt{P},B}\left(G_\mu\right)\right\rangle\Big|_{\neg B} \middle| k \in \mathbb{N} \right\}$ for $H_{\mathtt{P},B} : K \mapsto [\![\mathtt{P}]\!]\left(\langle K\rangle|_B\right) + \langle K\rangle|_{\neg B}$.

We can divide $H^k_{\mathtt{P},B}\left(G_\mu\right)$ into two parts: One which contains all terms which are not part of the PGF restricted to B and the other part which is modified by further executions of the while-loop.

The part of the PGF which is further modified is the non-terminating term.

Definition 4.1.1 (Non-Terminating Term)**.**
The non-terminating term ${}_{\mathtt{P},G_\mu,B}n_k$ is for all $k \in \mathbb{N}$ defined as follows:

$$
\begin{aligned}
{}_{\mathtt{P},G_\mu,B}n_0 &= \langle G_\mu\rangle|_B \\
{}_{\mathtt{P},G_\mu,B}n_{k+1} &= \left\langle [\![\mathtt{P}]\!]\left({}_{\mathtt{P},G_\mu,B}n_k\right)\right\rangle\Big|_B
\end{aligned}
$$

△

The other part is the resulting term which contains only terms which are not part of the input PGF restricted to B.

Definition 4.1.2 (Resulting Term)**.**
The resulting term ${}_{\mathtt{P},G_\mu,B}e_k$ is for all $k \in \mathbb{N}$ defined as follows:

$$\begin{aligned} {}_{\mathtt{P},G_\mu,B}e_0 &= \langle G_\mu \rangle|_{\neg B} \\ {}_{\mathtt{P},G_\mu,B}e_{k+1} &= \left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle\Big|_{\neg B} + {}_{\mathtt{P},G_\mu,B}e_k \end{aligned}$$

△

The sum of those two terms always equals $H^k_{\mathtt{P},B}(G_\mu)$.

Lemma 4.1.3. *${}_{P,G_\mu,B}e_k + {}_{P,G_\mu,B}n_k = H^k_{P,B}(G_\mu)$ for all $k \in \mathbb{N}$.*

The proof is done by induction over $k \in \mathbb{N}$ and can be found in the appendix (page **??**).

In addition, ${}_{\mathtt{P},G_\mu,B}e_k = \left\langle H^k_{\mathtt{P},B}(G_\mu) \right\rangle\Big|_{\neg B}$ for all $k \in \mathbb{N}$. This is important so that we can relate the definition of $[\![\mathtt{while}(B)\{\mathtt{P}\}]\!]$ to $\left\{ {}_{\mathtt{P},G_\mu,B}e_k \middle| k \in \mathbb{N} \right\}$.

Lemma 4.1.4. *${}_{P,G_\mu,B}e_k = \left\langle H^k_{P,B}(G_\mu) \right\rangle\Big|_{\neg B}$ for all $k \in \mathbb{N}$.*

Proof. We will prove this directly with the previous lemma.

$$\begin{aligned} \left\langle H^k_{\mathtt{P},B}(G_\mu) \right\rangle\Big|_{\neg B} &= \left\langle {}_{\mathtt{P},G_\mu,B}e_k + {}_{\mathtt{P},G_\mu,B}n_k \right\rangle\Big|_{\neg B} && \text{(by Lemma 4.1.3)} \\ &= \left\langle {}_{\mathtt{P},G_\mu,B}e_k \right\rangle\Big|_{\neg B} + \left\langle {}_{\mathtt{P},G_\mu,B}n_k \right\rangle\Big|_{\neg B} \\ & && \text{(restriction is linear (Theorem 3.4.2))} \\ &= {}_{\mathtt{P},G_\mu,B}e_k && \left(\left\langle {}_{\mathtt{P},G_\mu,B}n_k \right\rangle\Big|_{\neg B} = 0,\ \left\langle {}_{\mathtt{P},G_\mu,B}e_k \right\rangle\Big|_{\neg B} = {}_{\mathtt{P},G_\mu,B}e_k \right)^1 \end{aligned}$$

□

Therefore, all the properties which hold for $\left\{ \left\langle H^k_{\mathtt{P},B}(G_\mu) \right\rangle\Big|_{\neg B} \middle| k \in \mathbb{N} \right\}$ also hold for $\left\{ {}_{\mathtt{P},G_\mu,B}e_k \middle| k \in \mathbb{N} \right\}$.

Corollary 4.1.5.

$$[\![\mathtt{while}(B)\{\mathtt{P}\}]\!](G_\mu) = \bigsqcup \left\{ {}_{\mathtt{P},G_\mu,B}e_k \middle| k \in \mathbb{N} \right\}.$$

In addition, $\left\{ {}_{\mathtt{P},G_\mu,B}e_k \middle| k \in \mathbb{N} \right\}$ is an ω-chain.

This directly follows from Definition 3.3.5, Lemma 3.3.7 and Lemma 4.1.4.

We can define the resulting terms only by the sum of $[\![\mathtt{P}]\!]$ applied to non-terminating terms restricted to $\neg B$.

[1] Restriction is idempotent.

Lemma 4.1.6. ${}_{\mathtt{P},G_\mu,B}e_k = \sum_{i=0}^{k-1} \left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_i \right) \right\rangle \Big|_{\neg B} + {}_{\mathtt{P},G_\mu,B}e_0$ *for all* $k \in \mathbb{N}$.

The proof is done by induction over $k \in \mathbb{N}$ and can be found in the appendix (page **??**).

With that definition in mind we can formulate a rule. This rule says that whenever the terms, namely $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle \Big|_{\neg B}$, are of a certain form we can deduce the form of the result of the while-loop.

Theorem 4.1.7. *If* $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle \Big|_{\neg B} = b \cdot a^k$ *for all* $k \in \mathbb{N}$ *and two constants* $a, b \in L$ *then*

$$[\![\mathtt{while}(B)\{\mathtt{P}\}]\!](G_\mu) = \frac{b}{1-a} + {}_{\mathtt{P},G_\mu,B}e_0.$$

Proof. Because of Lemma 4.1.6 ${}_{\mathtt{P},G_\mu,B}e_k = \sum_{i=0}^{k-1} \left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_i \right) \right\rangle \Big|_{\neg B} + {}_{\mathtt{P},G_\mu,B}e_0$. Since $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle \Big|_{\neg B} = b \cdot a^k$ for all $k \in \mathbb{N}$ and two constants b, a,

$${}_{\mathtt{P},G_\mu,B}e_k = \sum_{i=0}^{k-1} \left(b \cdot a^i \right) + {}_{\mathtt{P},G_\mu,B}e_0 = b \sum_{i=0}^{k-1} \left(a^i \right) + {}_{\mathtt{P},G_\mu,B}e_0$$

By Corollary 4.1.5 $[\![\mathtt{while}(B)\{\mathtt{P}\}]\!](G_\mu) = \bigsqcup \left\{ {}_{\mathtt{P},G_\mu,B}e_k \middle| k \in \mathbb{N} \right\}$. The sequence $\left\{ {}_{\mathtt{P},G_\mu,B}e_k \middle| k \in \mathbb{N} \right\}$ has a supremum. In addition, it is monotonic since it is an ω-chain (Lemma 4.1.5). Sequences which are monotonic converge (Lemma 3.1.7). The limit is

$$\lim_{k \to \infty} {}_{\mathtt{P},G_\mu,B}e_k = b \sum_{i=0}^{\infty} a^i + {}_{\mathtt{P},G_\mu,B}e_0.$$

With the formula for geometric series, we can find a closed form for that expression. This closed form is

$$b \sum_{i=0}^{\infty} a^i + {}_{\mathtt{P},G_\mu,B}e_0 = \frac{b}{1-a} + {}_{\mathtt{P},G_\mu,B}e_0.$$

Since $\left\{ {}_{\mathtt{P},G_\mu,B}e_k \middle| k \in \mathbb{N} \right\}$ has a limit, the limit equals the supremum and

$$[\![\mathtt{while}(B)\{\mathtt{P}\}]\!](G_\mu) = \frac{b}{1-a} + {}_{\mathtt{P},G_\mu,B}e_0.$$

□

Example 4.1.8. This proof rule can be applied to the program we have seen several times before:

Program 3

```
1: x:=0;
2: {c:=0} [0.5] {c:=1}
3: while (c≠0) {
4:         x:=x+1;
5:         {c:=0} [0.5] {c:=1}
6: }
```

For that program, the following holds

$$ {}_{\mathtt{P},G_\mu,B}n_k = 0.5^{k+1}X^kC^1 $$

This can be shown by induction over $k \in \mathbb{N}$.

Proof. By induction over $k \in \mathbb{N}$.

Basis: $k = 0$

$$ {}_{\mathtt{P},G_\mu,B}n_0 = \langle G_\mu \rangle|_B = 0.5X^0C^1 = 0.5^{0+1}X^0C^1 $$

Induction Hypothesis: ${}_{\mathtt{P},G_\mu,B}n_k = 0.5^{k+1}X^kC^1$ for an arbitrary, but fixed $k \in \mathbb{N}$.

Inductive Step: $k \to k+1$

$$ \begin{aligned} {}_{\mathtt{P},G_\mu,B}n_{k+1} &= \left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle \Big|_B \\ &= \left\langle [\![\mathtt{P}]\!] \left(0.5^{k+1}X^kC^1 \right) \right\rangle \Big|_B \qquad \text{(IH)} \\ &= 0.5^{k+2}X^{k+1}C^1 \end{aligned} $$

□

With that in mind we can deduce the form for $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle \Big|_{\neg B}$ for all $k \in \mathbb{N}$.

$$ \begin{aligned} \left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle \Big|_{\neg B} &= \left\langle [\![\mathtt{P}]\!] \left(0.5^{k+1}X^kC^1 \right) \right\rangle \Big|_{\neg B} \\ &= \left\langle 0.5^{k+2}X^{k+1}C^1 + 0.5^{k+2}X^{k+1}C^0 \right\rangle \Big|_{\neg B} \\ &= 0.5^{k+2}X^{k+1}C^0 \end{aligned} $$

We can set $b = 0.5^2C^0X^1$ and $a = 0.5X$ and have

$$ \left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle \Big|_{\neg B} = 0.5^2C^0X^1 \cdot 0.5^kX^k = 0.5^2C^0X^1 \cdot (0.5X)^k = b \cdot a^k $$

With our proof rule, Theorem 4.1.7, we can therefore deduce that

$$\begin{aligned}
[\![\texttt{while}(B)\{\texttt{P}\}]\!](G_\mu) &= \frac{b}{1-a} + {}_{\texttt{P},G_\mu,B}e_0 \\
&= \frac{0.5^2 X}{1-0.5X} + 0.5 \\
&= \frac{X}{4-2X} + \frac{2-X}{4-2X} \\
&= \frac{2}{4-2X} \\
&= \frac{1}{2-X}
\end{aligned}$$

△

We see that obtaining the result of a while-loop is still a lot of effort: We have to do one induction. After that, we use the result of the induction to get b and a and then, we can apply the proof rule. The main contribution of the proof rule is that we no longer have to do the analysis in order to obtain the supremum.

The proof rule works fine for programs like Program 3. However, there are more complex programs than this.

Example 4.1.9. Next, we look at the following program:

Program 4

```
1: x:=0; c:=1; t:=0;
2: while (c≠0) {
3:         if (t=0) {
4:                 {c:=0} [a] {t:=1}
5:         } else {
6:                 {c:=0} [b] {t:=0}
7:         };
8:         x:=x+1;
9: }
```

This program models the duel of two cowboys. Either it is the turn of Cowboy 0 or Cowboy 1. The cowboys hit the other one with probability a respectively probability b. If a cowboy hits the other one, the duel is over. Otherwise, it is the turn of the other cowboy. The variable x counts the duration of the duel.

We can look at some of the ${}_{\texttt{P},G_\mu,B}n_k$ to see whether we can deduce any regularities:

$$\begin{aligned}
{}_{\texttt{P},G_\mu,B}n_0 &= X^0 C^1 T^0 \\
{}_{\texttt{P},G_\mu,B}n_1 &= (1-a) X^1 C^1 T^1
\end{aligned}$$

$$
\begin{aligned}
{}_{\mathtt{P},G_\mu,B}n_2 &= (1-b)(1-a)X^2C^1T^0 \\
{}_{\mathtt{P},G_\mu,B}n_3 &= (1-a)(1-b)(1-a)X^3C^1T^1
\end{aligned}
$$

We see that we have two turns regarding the non-terminating term:

$$
\begin{aligned}
{}_{\mathtt{P},G_\mu,B}n_k &= \left[(1-b)(1-a)\right]^{\lfloor \frac{k}{2} \rfloor} X^k C^1 T^0 && \text{for } k\%2 = 0 \\
{}_{\mathtt{P},G_\mu,B}n_k &= (1-a)\left[(1-b)(1-a)\right]^{\lfloor \frac{k}{2} \rfloor} X^k C^1 T^1 && \text{for } k\%2 = 1
\end{aligned}
$$

Here, % is the modulo operator. We need to prove that form of the non-terminating term.

Proof. By induction over $k \in \mathbb{N}$

Basis: $k = 0$ and $k = 1$

$$
\begin{aligned}
{}_{\mathtt{P},G_\mu,B}n_0 &= \langle G_\mu \rangle|_B = X^0C^1T^0 = [(1-b)(1-a)]^{\lfloor \frac{0}{2} \rfloor} X^0C^1T^0 \\
{}_{\mathtt{P},G_\mu,B}n_1 &= \left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_0 \right) \right\rangle \Big|_B \\
&= \left\langle [\![\mathtt{P}]\!] \left(X^0C^1T^0 \right) \right\rangle \Big|_B \\
&= (1-a)X^1C^1T^1 \\
&= (1-a)[(1-b)(1-a)]^{\lfloor \frac{1}{2} \rfloor} X^1C^1T^1
\end{aligned}
$$

Induction Hypothesis: For an arbitrary, but fixed $k \in \mathbb{N}$, the following holds

$$
\begin{aligned}
{}_{\mathtt{P},G_\mu,B}n_k &= [(1-b)(1-a)]^{\lfloor \frac{k}{2} \rfloor} X^k C^1 T^0 && \text{for } k\%2 = 0 \\
{}_{\mathtt{P},G_\mu,B}n_k &= (1-a)[(1-b)(1-a)]^{\lfloor \frac{k}{2} \rfloor} X^k C^1 T^1 && \text{for } k\%2 = 1.
\end{aligned}
$$

Inductive Step: $k \to k+1$

There are two cases: $k\%2 = 0$ and $k\%2 = 1$.

1. $k\%2 = 0$

$$
\begin{aligned}
{}_{\mathtt{P},G_\mu,B}n_{k+1} &= \left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle \Big|_B \\
&= \left\langle [\![\mathtt{P}]\!] \left([(1-b)(1-a)]^{\lfloor \frac{k}{2} \rfloor} X^k C^1 T^0 \right) \right\rangle \Big|_B && \text{(IH)} \\
&= (1-a)[(1-b)(1-a)]^{\lfloor \frac{k}{2} \rfloor} X^{k+1} C^1 T^1 \\
&= (1-a)[(1-b)(1-a)]^{\lfloor \frac{k+1}{2} \rfloor} X^{k+1} C^1 T^1 && \text{(k\%2=0)}
\end{aligned}
$$

2. $k\%2 = 1$

$$\begin{aligned}
{}_{\mathtt{P},G_\mu,B}n_{k+1} &= \left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle \Big|_B \\
&= \left\langle [\![\mathtt{P}]\!] \left((1-a)[(1-b)(1-a)]^{\lfloor \frac{k}{2} \rfloor} X^k C^1 T^1 \right) \right\rangle \Big|_B && \text{(IH)} \\
&= (1-b)(1-a)[(1-b)(1-a)]^{\lfloor \frac{k}{2} \rfloor} X^{k+1} C^1 T^0 \\
&= [(1-b)(1-a)]^{\lfloor \frac{k+1}{2} \rfloor} X^{k+1} C^1 T^1 && \text{(k\%2=1)}
\end{aligned}$$

□

With that in mind, for $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle \Big|_{\neg B}$, the following holds:

$$\begin{aligned}
\text{k\%2=0} : \left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle \Big|_{\neg B} &= \left\langle [\![\mathtt{P}]\!] \left([(1-b)(1-a)]^{\lfloor \frac{k}{2} \rfloor} X^k C^1 T^0 \right) \right\rangle \Big|_{\neg B} \\
&= a[(1-b)(1-a)]^{\lfloor \frac{k}{2} \rfloor} X^{k+1} C^0 T^0 \\
\text{k\%2=1} : \left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle \Big|_{\neg B} &= \left\langle [\![\mathtt{P}]\!] \left((1-a)[(1-b)(1-a)]^{\lfloor \frac{k}{2} \rfloor} X^k C^1 T^1 \right) \right\rangle \Big|_{\neg B} \\
&= b(1-a)[(1-b)(1-a)]^{\lfloor \frac{k}{2} \rfloor} X^{k+1} C^0 T^1
\end{aligned}$$

△

For that case, it would be quite useful to have a proof rule. If we had one, we could now directly tell, what the resulting PGF would be. In fact, we developed a proof rule for these cases. It allows that the loop has a finite number of turns and tells what the result of that loop would be. If we have a specific number of turns t such that we can divide $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle \Big|_{\neg B}$ in three parts, $a, b, c \in L$ such that a develops according to the number of loop executions, b is a constant and c increases with the number of executions of a specific turn, then we have a closed form for $[\![\mathtt{while}(B)\{\mathtt{P}\}]\!](G_\mu)$.

Theorem 4.1.10. *If there exists $t \in \mathbb{N}^{\geq 1}$ and there exist $a_0, ..., a_{t-1}, b_0, ..., b_{t-1}, c_0, ..., c_{t-1} \in L$ such that for all $k \in \mathbb{N}$*

$$\begin{aligned}
\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle \Big|_{\neg B} &= b_{k\%t} a_{k\%t}^k c_{k\%t}^{\lfloor \frac{k}{t} \rfloor} \\
\text{then } [\![\mathtt{while}(B)\{\mathtt{P}\}]\!](G_\mu) &= \sum_{j=0}^{t-1} \frac{b_j a_j^j}{1 - a_j^t c_j} + {}_{\mathtt{P},G_\mu,B}e_0.
\end{aligned}$$

Proof. Because of Lemma 4.1.6 and the fact that $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle \Big|_{\neg B} = b_{k\%t} a_{k\%t}^k c_{k\%t}^{\lfloor \frac{k}{t} \rfloor}$,

$${}_{\mathtt{P},G_\mu,B}e_k = \sum_{i=0}^{k-1} \left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_i \right) \right\rangle \Big|_{\neg B} + {}_{\mathtt{P},G_\mu,B}e_0$$

$$= \sum_{j=0}^{t-1} \left(\sum_{l \in \{x | x\%t = j \wedge x \leq k-1\}} \left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B} n_l \right) \right\rangle \Big|_{\neg B} \right) + {}_{\mathtt{P},G_\mu,B} e_0.$$

The two expressions for ${}_{\mathtt{P},G_\mu,B} e_k$ differ only in the ordering of their terms: In the first expression, the index of $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B} n_k \right) \right\rangle \Big|_{\neg B}$ is increasing continuously. For the second expression the terms are grouped by turns. We first sum up all $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B} n_k \right) \right\rangle \Big|_{\neg B}$, which belong to turn 0 and add to that value the sum of all $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B} n_k \right) \right\rangle \Big|_{\neg B}$ which belong to turn 1 and so on. In order to sum up only the terms which belong to a certain turn j, we have the condition $x\%t = j$.

Now, we look at the different subsums of ${}_{\mathtt{P},G_\mu,B} e_k$ and define:

$$m_{j_k} = \sum_{l \in \{x | x\%t = j \wedge x \leq k-1\}} \left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B} n_l \right) \right\rangle \Big|_{\neg B} \qquad \text{for all } j \in \{0, ...t-1\}.$$

With that definition, we have

$$\begin{aligned} {}_{\mathtt{P},G_\mu,B} e_k &= \sum_{j=0}^{t-1} \sum_{l \in \{x | x\%t = j \wedge x \leq k-1\}} \left(\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B} n_l \right) \right\rangle \Big|_{\neg B} \right) + {}_{\mathtt{P},G_\mu,B} e_0 \\ &= \sum_{j=0}^{t-1} \left(m_{j_k} \right) + {}_{\mathtt{P},G_\mu,B} e_0 \end{aligned}$$

We can change the form of m_{j_k} in the following way:

$$\begin{aligned} m_{j_k} &= \sum_{l \in \{x | x\%t = j \wedge x \leq k-1\}} \left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B} n_l \right) \right\rangle \Big|_{\neg B} \\ &= \sum_{l \in \{x | x\%t = j \wedge x \leq k-1\}} b_{l\%t} a_{l\%t}^l c_{l\%t}^{\lfloor \frac{l}{t} \rfloor} && \text{(precondition)} \\ &= \sum_{l \in \{x | x\%t = j \wedge x \leq k-1\}} b_j a_j^l c_j^{\lfloor \frac{l}{t} \rfloor} && (l\%t = j) \\ &= b_j \sum_{l \in \{x | x\%t = j \wedge x \leq k-1\}} a_j^l c_j^{\lfloor \frac{l}{t} \rfloor} \\ &= b_j \sum_{\substack{l \in \{x | x = i \cdot t + j \text{ for all } i \in \mathbb{N} \\ \wedge x \leq k-1\}}} a_j^l c_j^{\lfloor \frac{l}{t} \rfloor}. && (x\%t = j \Rightarrow \exists i \in \mathbb{N} : i \cdot t + j = x) \end{aligned}$$

In order to modify the description of the set $\{x | x = i \cdot t + j \text{ for any } i \in \mathbb{N} \wedge x \leq k-1\}$ a bit more, we can transform the inequality $x \leq k-1$.

$$x \leq k - 1$$

$$
\begin{aligned}
&\Leftrightarrow && i \cdot t + j \leq k - 1 \\
&\Leftrightarrow && i \cdot t \leq k - 1 - j \\
&\Leftrightarrow && i \leq \frac{k-1-j}{t} \\
&\Leftrightarrow && i \leq \left\lfloor \frac{k-1-j}{t} \right\rfloor && (i \text{ is a natural number})
\end{aligned}
$$

Now, we have:

$$
\begin{aligned}
m_{j_k} &= b_j \sum_{\substack{l \in \{x \mid x = i \cdot t + j \\ \wedge x \leq k-1 \ \ i \in \mathbb{N}\}}} a_j^l c_j^{\left\lfloor \frac{l}{t} \right\rfloor} \\
&= b_j \sum_{\substack{l \in \{x \mid x = i \cdot t + j \\ \wedge i \leq \left\lfloor \frac{k-1-j}{t} \right\rfloor\}}} a_j^l c_j^{\left\lfloor \frac{l}{t} \right\rfloor} \\
&= b_j \sum_{i=0}^{\left\lfloor \frac{k-1-j}{t} \right\rfloor} a_j^{i \cdot t + j} c_j^{\left\lfloor \frac{i \cdot t + j}{t} \right\rfloor}
\end{aligned}
$$

We can transform $\left\lfloor \frac{i \cdot t + j}{t} \right\rfloor$:

$$
\left\lfloor \frac{i \cdot t + j}{t} \right\rfloor = \left\lfloor \frac{i \cdot t}{t} + \frac{j}{t} \right\rfloor = \left\lfloor i + \frac{j}{t} \right\rfloor \overset{j \leq t-1}{=} i.
$$

Now, we have:

$$
m_{j_k} = b_j \sum_{i=0}^{\left\lfloor \frac{k-1-j}{t} \right\rfloor} a_j^{i \cdot t + j} c_j^{\left\lfloor \frac{i \cdot t + j}{t} \right\rfloor} = b_j \sum_{i=0}^{\left\lfloor \frac{k-1-j}{t} \right\rfloor} a_j^{i \cdot t + j} c_j^i
$$

$\{m_{j_k}\}_{k \in \mathbb{N}}$ is monotonic. This is due to the fact that with increasing $k \in \mathbb{N}$ we only add more $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P}, G_\mu, B} n_k \right) \right\rangle \Big|_{\neg B}$ to the previous m_{j_k}. Monotonic sequences have a limit (Lemma 3.1.7). This is

$$
\lim_{k \to \infty} m_{j_k} = b_j \sum_{i=0}^{\infty} a_j^{i \cdot t + j} c_j^i.
$$

Since $\left\{ {}_{\mathtt{P}, G_\mu, B} e_k \middle| k \in \mathbb{N} \right\}$ is monotonic, it has a supremum which is the limit of the sequence (Lemma 3.1.7). So we conclude:

$$
\begin{aligned}
[\![\mathtt{while}(B)\{\mathtt{P}\}]\!] (G_\mu) &= \bigsqcup \left\{ {}_{\mathtt{P}, G_\mu, B} e_k \middle| k \in \mathbb{N} \right\} \\
&= \lim_{k \to \infty} {}_{\mathtt{P}, G_\mu, B} e_k
\end{aligned}
$$

$$
\begin{aligned}
&= \lim_{k\to\infty} \sum_{j=0}^{t-1} m_{j_k} + _{\mathtt{P},G_\mu,B} e_0 \\
&= \sum_{j=0}^{t-1} \lim_{k\to\infty} m_{j_k} + _{\mathtt{P},G_\mu,B} e_0 \\
&= \sum_{j=0}^{t-1} b_j \sum_{i=0}^{\infty} a_j^{i\cdot t+j} c_j^i + _{\mathtt{P},G_\mu,B} e_0 \\
&= \sum_{j=0}^{t-1} b_j \sum_{i=0}^{\infty} a_j^j \cdot a_j^{i\cdot t} c_j^i + _{\mathtt{P},G_\mu,B} e_0 \\
&= \sum_{j=0}^{t-1} b_j a_j^j \sum_{i=0}^{\infty} \left(a_j^t c_j\right)^i + _{\mathtt{P},G_\mu,B} e_0 \\
&= \sum_{j=0}^{t-1} \frac{b_j a_j^j}{1-a_j^t c_j} + _{\mathtt{P},G_\mu,B} e_0
\end{aligned}
$$

□

Example 4.1.11. With that rule, we can continue the analysis of Program 4. We found out so far that

$$
\begin{aligned}
k\%2 = 0 : \left\langle [\![\mathtt{P}]\!] \left(_{\mathtt{P},G_\mu,B} n_k \right) \right\rangle \Big|_{\neg B} &= a[(1-b)(1-a)]^{\lfloor \frac{k}{2} \rfloor} X^{k+1} C^0 T^0 \\
k\%2 = 1 : \left\langle [\![\mathtt{P}]\!] \left(_{\mathtt{P},G_\mu,B} n_k \right) \right\rangle \Big|_{\neg B} &= b(1-a)[(1-b)(1-a)]^{\lfloor \frac{k}{2} \rfloor} X^{k+1} C^0 T^1
\end{aligned}
$$

So the preliminaries of the second proof rule, Theorem 4.1.10 are fulfilled and the relevant parameters are given by

$$
\begin{aligned}
t = 2,\ a_0 = X,\ b_0 = aXC^0T^0,\ c_0 = (1-b)(1-a), \\
a_1 = X,\ b_1 = b(1-a)XC^0T^1,\ c_1 = (1-b)(1-a)
\end{aligned}
$$

So according to the rule

$$
\begin{aligned}
[\![\mathtt{while}(B)\{\mathtt{P}\}]\!](G_\mu) &= \sum_{j=0}^{t-1} \frac{b_j a_j^j}{1-a_j^t c_j} + _{\mathtt{P},G_\mu,B} e_0 \\
&= \frac{aX}{1-X^2(1-b)(1-a)} + \frac{b(1-a)X^2T}{1-X^2(1-b)(1-a)} \\
&= \frac{aX + b(1-a)X^2T}{1-X^2(1-b)(1-a)}
\end{aligned}
$$

△

4.2 Applications for the Proof Rules

In the previous section, we introduced two proof rules. In the second part of this section, we look at what type of programs we can apply those rules to. In this section we study one form of program which is based on permutations. Therefore, the first part of this section will be an introduction to permutations.

Introduction to Permutations and Permutation Matrices

In the following we will introduce permutations and propose some statements about permutations.

Definition 4.2.1 (Permutation [**?**])**.**
A permutation π over a finite set is a bijective mapping on that set. A permutation $\pi\colon \{a_1, ..., a_k\} \to \{a_1, ..., a_k\}$ can be written as

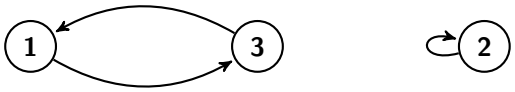
$$\pi = \begin{pmatrix} a_1 & \dots & a_n \\ \pi(a_1) & \dots & \pi(a_n) \end{pmatrix}$$

△

Example 4.2.2. Let π be a permutation, $\pi\colon \{1,2,3\} \to \{1,2,3\}$ with

$$\pi = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}.$$

Then we can visualise the permutation by a directed graph with the elements $1, 2$ and 3 as nodes and the effect of π as edges:



△

We can construct a matrix A_π corresponding to a permutation π. If we multiply this matrix with a vector the rows of the vector will be permuted according to the permutation π [**?**].

Definition 4.2.3 (Permutation Matrix [**?**])**.**
Let π be a permutation $\pi\colon \{1, ..., n\} \to \{1, ..., n\}$. Let $e^i = (0, .., 1, ..., 0)$ be the row vector which contains only zeros except for position i where it is one. Then we can construct

the permutation matrix

$$A_\pi = \begin{pmatrix} e^{\pi(1)} \\ \vdots \\ e^{\pi(n)} \end{pmatrix}.$$

△

Example 4.2.4. Let π be the permutation from Example 4.2.2,

$$\pi = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix},$$

then the permutation matrix is given by

$$A_\pi = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

If we multiply A_π with a vector we see that its rows are permuted according to π.

$$A_\pi \cdot \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \\ 2 \end{pmatrix}$$

△

If we multiply an n-th power of a permutation matrix with a vector the rows of the vector are n-times permuted according to π.

Lemma 4.2.5. *For all permutations π, the corresponding permutation matrix $A_\pi \in \mathbb{N}^{n \times n}$ and all vectors $\vec{x} \in \mathbb{N}^n$, the following holds*

$$A_\pi^k \vec{x} = \begin{pmatrix} x_{\pi^k(1)} \\ \vdots \\ x_{\pi^k(n)} \end{pmatrix}.$$

The proof is done by induction and can be found in the appendix (page **??**).

There are special permutations which are called *cycles*. We define those as follows:

Definition 4.2.6 (Cycle)**.**
A permutation $\pi\colon \{a_1, ..., a_k\} \to \{a_1, ..., a_k\}$ is a *cycle* if there exists an ordering $n_1, ..., n_k$ of the numbers $\{a_1, ..., a_k\}$ such that

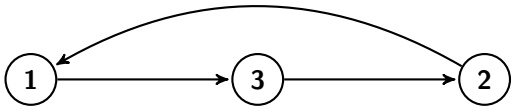
$$\begin{aligned} \pi(n_i) &= n_{i+1} && \text{for } 1 \leq i < k \\ \pi(n_k) &= n_1. \end{aligned}$$

We can denote a cycle $\pi\colon \{a_1, ..., a_k\} \to \{a_1, ..., a_k\}$ as $\pi = (n_1 \ ... \ n_k)$. △

Example 4.2.7. We now introduce a permutation which is a cycle, $\pi\colon \{1, 2, 3\} \to \{1, 2, 3\}$ with

$$\pi = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix} \text{ or } \pi = (1\ 3\ 2)\,.$$

We see that we can visualise that by



So for $n_1 = 1, n_2 = 3, n_3 = 2$, the following holds

$$\begin{aligned} \pi(n_i) &= n_{i+1} && \text{for } 1 \leq i < 3 \\ \pi(n_3) &= n_1. \end{aligned}$$

△

If we apply a cycle k times to an element the result is the same element.
Lemma 4.2.8. *For all cycles* $\pi\colon \{a_1, ..., a_k\} \to \{a_1, ..., a_k\}$*, the following holds*

$$\pi^k(a_i) = a_i, \qquad \text{for all } i \in \{1, ..., k\}.$$

The proof can be found in the appendix (page **??**).

In fact, we can generalize that lemma and say that if we apply the permutation a multiple of the length of the cycle, we still get the original element.
Lemma 4.2.9. *For all cycles* $\pi\colon \{a_1, ..., a_k\} \to \{a_1, ..., a_k\}$ *and all* $l \in \mathbb{N}$*, the following holds*

$$\pi^{l \cdot k}(a_i) = a_i, \qquad \text{for all } i \in \{1, ..., k\}.$$

The proof is done by induction and can be found in the appendix (page **??**).

We can divide any permutation uniquely into cycles. We can already see that in the graph of Example 4.2.2. The graph has two circles which correspond to the cycles the permutation can be decomposed into.

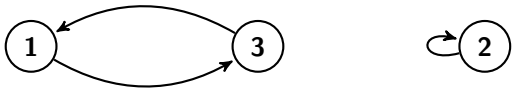
Lemma 4.2.10 ([**?**])**.** *Every permutation π can be decomposed into distinct cycles $c_1, ..., c_m$.*

The proof is done by constructing an algorithm which finds the different cycles [**?**].

Example 4.2.11. We can decompose the permutation from Example 4.2.2, $\pi\colon \{1,2,3\} \to \{1,2,3\}$ with

$$\pi = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$$

into two cycles: $c_1 = (1\ 3)$, $c_2 = (2)$ and write $\pi = (1\ 3)(2)$. With the visualisation, we see that the graph representing π has two circles which correspond to the two cycles of π.



△

In the following for a vector $\vec{y} \in \mathbb{N}^n, \vec{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$, y_i denotes the i-th component of the vector.

Now, we introduce some notations:

Definition 4.2.12 (Notations for Permutations)**.**

We introduce several notations to refer to specific cycles of a permutation. Let $\pi\colon \{1, ..., n\} \to \{1, ..., n\}$ be a permutation, $A_\pi \in \mathbb{N}^{n \times n}$ be the permutation matrix of π and $\vec{d} \in \mathbb{N}^n$ be a vector.

- For a cycle c, we define $set(c)$ as a set containing all elements the cycle is defined on. So if we have a cycle $c\colon \{a_1, ..., a_k\} \to \{a_1, ..., a_k\}$, $set(c) = \{a_1, ..., a_k\}$.
- Every permutation $\pi\colon \{1, ..., n\} \to \{1, ..., n\}$ can be decomposed into distinct cycles $c_1, .., c_m$. We number the cycles. The numbering works as follows:
  - Each cycle can be uniquely identified by its least element, $\min(set(c))$, it is defined on.
  - We sort the least elements of all cycles in a tuple $s = (s_1, ..., s_m)$ such that any s_i is a least element of a cycle and for any cycle the least element is a component of the tuple. In addition, the tuple is sorted such that $s_i < s_{i+1}$

for all $i \in \{1, ..., m-1\}$.

– With the expression c_i we refer to the cycle whose least element is s_i. So, c_i is the cycle c for which $\min(set(c)) = s_i$ holds.

- $c_{(i)}$ is the cycle c of a permutation π such that $i \in set(c)$. This is unique because of Lemma 4.2.10.

- $len(c)$: The length of a cycle is defined as the size of its set such that $len(c) = |set(c)|$.

- $cyclesum(i)$: If we have a function of the form $f(\vec{x}) = A_\pi \vec{x} + \vec{d}$, the $cyclesum(i)$ is the sum of all coefficients of $\vec{d}$ which are in the set of the cycle $c_{(i)}$,

$$cyclesum(i) = \sum_{j \in set\left(c_{(j)}\right)} d_j = \sum_{j=0}^{len\left(c_{(i)}\right)-1} d_{\pi^j(i)}.$$

- If it is important to relate a cycle not only to an element but to a permutation π, the notations $cycle_{\pi,i}$, $cycle_{\pi,(i)}$ and $cyclesum_\pi(i)$ are used.

△

Example 4.2.13. To be reassured regarding the different notations, we will use these notations for the permutation from the previous examples, Example 4.2.2 and Example 4.2.11, $\pi = (1\ 3)(2)$ and

$$f\colon \mathbb{N}^3 \to \mathbb{N}^3, \vec{x} \mapsto \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \vec{x} + \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}.$$

Then the notation works as follows:

- $set\,((1\ 3)) = \{1, 3\}$ and $set\,((2)) = \{2\}$

- $c_1 = (1\ 3)$ and $c_2 = (2)$ since the least elements of the cycles are 1 and 2.

- $c_{(3)} = (1\ 3)$

- $len(c_1) = |set(c_1)| = |\{1, 3\}| = 2$ and $len(c_2) = |set(c_2)| = |\{2\}| = 1$

- $cyclesum(1) = \sum_{j=0}^{len\left(c_{(1)}\right)-1} d_{\pi^j(1)} = \sum_{j=0}^{1} d_{\pi^j(1)} = 2 + 3 = 5$

△

While-Loops and Permutations

The second proof rule, Theorem 4.1.10, makes a statement about the PGF resulting from a while-loop if the resulting terms have a certain form. We look at a form of a while-loop such that the resulting terms have the form required for the use of Theorem 4.1.10.

Now and in the following, we have the vectors $\vec{x}, \vec{d} \in \mathbb{N}^n$, $\vec{p} \in \mathbb{N}^l$, the permutations $\pi\colon \{1,...,n\} \to \{1,...,n\}$ and $\sigma\colon \{1,...,l\} \to \{1,...,l\}$ and the permutation matrices A_π and B_σ. In addition, the function f is defined as $f\colon \mathbb{N}^n \to \mathbb{N}^n, \vec{x} \mapsto A_\pi \vec{x} + \vec{d}$. We will show that for all while-loops of the following form, Theorem 4.1.10 can be used:

Program 5

```
while (c ≠ 0) {
        x⃗ := A_π · x⃗ + d⃗;
        p⃗ := B_σ · p⃗;
        {c := 0}[p_1]{c := 1}
}
```

The requirements of Theorem 4.1.10 were that

$$\exists t \in \mathbb{N}^{\geq 1} : \exists a_0, ..., a_{t-1}, b_0, ..., b_{t-1}, c_0, ..., c_{t-1} \in L :$$
$$\forall k \in \mathbb{N} : \left\langle [\![\mathtt{P}]\!] \left(_{\mathtt{P},G_\mu,B} n_k\right)\right\rangle\Big|_{\neg B} = b_{k\%t} a_{k\%t}^k c_{k\%t}^{\lfloor \frac{k}{t} \rfloor}.$$

We can structure the non-terminating and the resulting terms in exactly as many turns as the product of length of all cycles of π and σ, $t = len\,(c_{\sigma,1}) \prod_{i=1}^{m} \left(len\,(c_{\pi,i}) \right)$ where m is the number of cycles of π. Before we start proving that, we illustrate this by an example.

Example 4.2.14. We look at the program

Program 6

```
c := 1; t := 0; a := 0; b := 0; e := 0;
while (c ≠ 0) {
        ⟨a := b + 2; b := e + 1; e := a + 3⟩;
        if (t = 0) {
                {c := 0}[g]{c := 1};
                t := 1
        } else {
                {c := 0}[h]{c := 1};
                t := 0
        }
}
```

The instruction in the brackets $\langle\rangle$ are done simultaneously. We can formulate this program such that it has the form of Program 5. We will ignore the value of t since it is only part of the program to switch between the different probabilities.

Program 7

$c := 1; \vec{x} := \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \vec{p} := \begin{pmatrix} h \\ g \end{pmatrix};$

`while` $(c \neq 0)$ {

$\vec{x} := \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \vec{x} + \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix};$

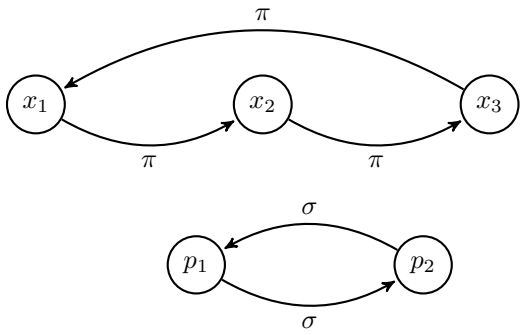
$\vec{p} := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \vec{p};$

$\{c := 0\}[p_1]\{c := 1\}$

}

Here, a from Program 6 corresponds to x_1, b to x_2 and e to x_3. In the following $\vec{d} = \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}$.

If we visualise the permutations π and σ, we get the following graphs:



In total we have two cycles based on π and σ. The product of the length of those two cycles is six. That means that after six applications of the permutation π and σ, the result is the original value for any component of $\vec{x}$ and $\vec{p}$. However, we also have to take into account that the vector $\vec{x}$ is summed up with $\vec{d}$ in all executions of the loop. The values added to the vector $\vec{x}$ during the six executions of the loop are always the same since $\vec{d}$ is constant. That means, we can compute the values added to a component of $\vec{x}$ during one turn of the program in advance and can deduce the form of ${}_{\mathtt{P},G_\mu,B}n_k$ respectively $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right\rangle\Big|_{\neg B}$ with respect to six turns. △

In the following, we want to prove that $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right\rangle\Big|_{\neg B}$ has the form required for

Theorem 4.1.10 for all $k \in \mathbb{N}$. To do so, we have to take steps and look first at the form of the non-terminating terms, ${}_{\mathtt{P},G_\mu,B}n_k$. When computing the non-terminating terms we have to know what the program does exactly. We will analyse that in the following.

The manipulation of the vector $\vec{x}$ in Program 5 can be described as a function f

$$f\colon \mathbb{N}^n \to \mathbb{N}^n,\ \vec{x} \mapsto A_\pi \vec{x} + \vec{d}.$$

We want to know what the components of $\vec{x}$ look like after we applied this function t times in order to describe ${}_{\mathtt{P},G_\mu,B}n_k$. This task is much easier if we had a closed form for $f^k(\vec{x})$. In fact, there exists such a closed form.

Lemma 4.2.15. *Let f be a function $f\colon \mathbb{N}^n \to \mathbb{N}^n,\ \vec{x} \mapsto A_\pi \vec{x} + \vec{d}$ with $\vec{d} \in \mathbb{N}^n$, $A_\pi \in \mathbb{N}^{n\times n}$ and A_π is a permutation matrix.*
Then

$$f^k(\vec{x}) = A_\pi^k \vec{x} + \sum_{i=0}^{k-1} A_\pi^i \vec{d} \qquad \text{for all } k \in \mathbb{N}.$$

The proof is done by induction over $k \in \mathbb{N}$ and can be found in the appendix (page **??**).

With that closed form, we can deduce the form of the components of $f^t(\vec{x})$, $\left(f^t(\vec{x})\right)_j$ for $j \in \{1,...,n\}$. After t applications of f, $\left(A_\pi^t \vec{x}\right)_j = x_j$ since t is a multiple of the length of the cycle $c_{\pi,(j)}$ (Lemma 4.2.9). The sum $\left(\sum_{i=0}^{t-1} A_\pi^i \vec{d}\right)_j$ can also be simplified as we will see.

Lemma 4.2.16. *Let f be a function $f\colon \mathbb{N}^n \to \mathbb{N}^n$, $f\colon \vec{x} \to A_\pi \cdot \vec{x} + \vec{d}$ where $\vec{x},\ \vec{d} \in \mathbb{N}^n$. A_π is the permutation matrix of $\pi\colon \{1,...,n\} \to \{1,...,n\}$.*
Then $\left(f^t(\vec{x})\right)_j = x_j + \frac{t}{len\left(c_{(j)}\right)} cyclesum(j)$ for all $j \in \{1,...,n\}$, $t = \prod_{i=1}^{m} len(c_i)$ and m is the number of cycles of π.

Proof. We first look at what happens to x_j if we apply f $len\left(c_{(j)}\right)$-times. In fact, the j-th component of the resulting vector, $\left(f^{len\left(c_{(j)}\right)}(\vec{x})\right)_j$, is the original j-th component of the input vector summed up with $cyclesum(j)$. $cyclesum(j)$ was the sum of all components of $\vec{d}$ which were part of *set* $\left(c_{(j)}\right)$, $cyclesum(j) = \sum_{i=0}^{len\left(c_{(j)}\right)-1} d_{\pi^i(j)}$.

First, we prove that

$$\left(f^{len\left(c_{(j)}\right)}(\vec{x})\right)_j = x_j + cyclesum(j) \qquad \text{for all } j \in \{1,...,n\}$$

$$\begin{aligned}
\left(f^{len\left(c_{(j)}\right)}\left(\vec{x}\right)\right)_j &= \left(A_\pi^{len\left(c_{(j)}\right)}\vec{x} + \sum_{i=0}^{len\left(c_{(j)}\right)-1} A_\pi^i \vec{d}\right)_j && \text{(Lemma 4.2.15)}\\
&= \left(A_\pi^{len\left(c_{(j)}\right)}\vec{x}\right)_j + \left(\sum_{i=0}^{len\left(c_{(j)}\right)-1} A_\pi^i \vec{d}\right)_j\\
&= x_{\pi^{len\left(c_{(j)}\right)}\left(c_{(j)}\right)} + \sum_{i=0}^{len\left(c_{(j)}\right)-1} \left(A_\pi^i \vec{d}\right)_j && \text{(Lemma 4.2.5)}\\
&= x_{\pi^{len\left(c_{(j)}\right)}\left(c_{(j)}\right)} + \sum_{i=0}^{len\left(c_{(j)}\right)-1} d_{\pi^i(j)} && \text{(Lemma 4.2.5)}\\
&= x_j + \sum_{i=0}^{len\left(c_{(j)}\right)-1} d_{\pi^i(j)} && \text{(Lemma 4.2.8)}\\
&= x_j + cyclesum(j) && \text{(Definition 4.2.12)}
\end{aligned}$$

However, we want to know how the result of $f^t\left(\vec{x}\right)$ looks like. t is by definition a multiple of all $len\left(c_{(j)}\right)$. So, we have to see how a component x_j looks like if we apply f on the vector $\vec{x}$ a multiple of $len\left(c_{(j)}\right)$ times. The term x_j does not change because we permute a multiple of length of $c_{(j)}$ (Lemma 4.2.9). We have seen that if we apply f $len\left(c_{(j)}\right)$ times to $\vec{x}$ we have to add the $cyclesum(j)$ to x_j. If we apply f a multiple of $len\left(c_{(j)}\right)$ times to $\vec{x}$ we have to add a multiple of $cyclesum(j)$ to x_j.

For all $l \in \mathbb{N}$, the following holds

$$f^{l\cdot len\left(c_{(j)}\right)}(\vec{x})_j = \vec{x}_j + l \cdot cyclesum(j). \tag{4.1}$$

The proof is done by induction and can be found in the appendix (page **??**).

For $\left(f^t\left(\vec{x}\right)\right)_j$ we therefore have:

$$\begin{aligned}
\left(f^t\left(x\right)\right)_j &= \left(f^{\frac{t}{len\left(c_{(j)}\right)}\cdot len\left(c_{(j)}\right)}\left(x\right)\right)_j && \left(t \text{ is multiple of } len\left(c_{(j)}\right)\right)\\
&\overset{(4.1)}{=} \vec{x}_j + \frac{t}{len\left(c_{(j)}\right)} \cdot cyclesum(j)
\end{aligned}$$

□

Before we continue our analysis of the loop of Program 5, we look at the effect the rule $[\![x_j := e]\!]$ has on a PGF. For a PGF, $[\![x_j := e]\!]$ means that the exponents of the unknown

variables are changed according to the evaluation function e:

$$[\![x_j := e]\!](G_\mu) = \sum_{i_1,...,i_k \in \mathbb{N}} \mu(i_1,...,i_k) X_1^{i_1} \cdots X_j^{e(i_1,...,i_k)} \cdots X_k^{i_k}.$$

If the assignment has the form $x_j := x_j + i$ for an $i \in \mathbb{N}$, $[\![x_j := x_j + i]\!](G_\mu)$ is the same as $X_j^i \cdot G_\mu$. We will demonstrate this with an example.
Example 4.2.17. Suppose $G_\mu = 0.5X^1Y^1 + 0.5X^4Y^2$. The PGF resulting from an application of $[\![x := x + 3]\!]$ is

$$[\![x := x + 3]\!](G_\mu) = 0.5X^{1+3}Y^1 + 0.5X^{4+3}Y^2 = 0.5X^4Y^1 + 0.5X^7Y^2.$$

The effect $[\![x := x + 3]\!]$ has on G_μ is the same as multiplying G_μ with X^3:

$$X^3 G_\mu = X^3 \left(0.5X^1Y^1 + 0.5X^4Y^2\right) = 0.5X^{1+3}Y^1 + 0.5X^{4+3}Y^2 = 0.5X^4Y^1 + 0.5X^7Y^2.$$

△

Now, we continue our analysis of the loop of Program 5. We have one variable, c, on which the decision whether the loop is continued or not is based on. This decision is done by probabilistic choice. The probability is changing according to a permutation σ. So we have different probabilities for continuing the loop and determine by σ on which probability the decision is based on. $\vec{x}$ is changed according to a function f. So, the components of $\vec{x}$ are all changed at the same time.

The program has the property that we can see parallels between the different executions of the loop body: After we executed the loop as often as the length of cycles of π and σ (t) and look at $\vec{x}$, we can see parallels to the value of $\vec{x}$ t executions before.

We will show that based on the number of turns, t, we can deduce what the $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle \Big|_{\neg B}$ look like. We do this in steps: We first show that ${}_{\mathtt{P},G_\mu,B}n_{k+t}$ and ${}_{\mathtt{P},G_\mu,B}n_k$ differ only by a factor c. After that we show what form ${}_{\mathtt{P},G_\mu,B}n_k$ has for all $k \in \mathbb{N}$ and finally, we prove a form for $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k \right) \right\rangle \Big|_{\neg B}$.
Lemma 4.2.18. *For the loop of Program 5, the following holds* ${}_{P,G_\mu,B}n_{k+t} = c \cdot {}_{P,G_\mu,B}n_k$ *with* $c = \left(\prod_{i=1}^{t} \left(1 - p_{\sigma^i(1)}\right) \right) \prod_{i=1}^{n} X_i^{cyclesum_\pi(i) \frac{t}{len\left(c_{\pi,(i)}\right)}}$ *and* $t = len\,(c_{\sigma,1}) \prod_{i=1}^{m} \left(len\,(c_{\pi,i}) \right)$ *where m is the number of cycles of π.*

Proof. We observe that the loop body consists of two different parts: $\mathtt{P}_{assign}$ is where $\vec{x}$ is updated and $\mathtt{P}_{prob}$ in which the probability is updated and a probabilistic choice is made. So $\mathtt{P}_{assign}$ corresponds to $\vec{x} := A_\pi \cdot \vec{x} + \vec{d}$ and $\mathtt{P}_{prob}$ corresponds to $\vec{p} := B_\sigma \cdot \vec{p}; \{c := 0\}[\vec{p}_1]\{c := 1\}$.

Then for all $k \in \mathbb{N}$, the following holds:

$$\begin{aligned}
{}_{\mathtt{P},G_\mu,B}n_{k+1} &= \left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right\rangle\Big|_{c\neq 0} \\
&= \left\langle [\![\mathtt{P}_{assign};\mathtt{P}_{prob}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right\rangle\Big|_{c\neq 0} \\
&= \left\langle [\![\mathtt{P}_{prob}]\!] \left([\![\mathtt{P}_{assign}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right)\right\rangle\Big|_{c\neq 0}
\end{aligned}$$

One part of $\mathtt{P}_{prob}$ is the probabilistic decision $\{c := 0\}[p_1]\{c := 1\}$. It means that with probability p_1, c is set to zero afterwards, with probability $(1 - p_1)$ it is set to one. The probability p_1 depends on the number of loop executions. For now, we just write p.

$$\begin{aligned}
&{}_{\mathtt{P},G_\mu,B}n_{k+1} \\
&= restrict [\![\mathtt{P}_{prob}]\!] \left([\![\mathtt{P}_{assign}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right) c \neq 0 \\
&= \left\langle p \cdot [\![c := 0]\!] \left([\![\mathtt{P}_{assign}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right) + (1-p)\,[\![c := 1]\!] \left([\![\mathtt{P}_{assign}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right)\right\rangle\Big|_{c\neq 0} \\
&= \left\langle p \cdot [\![c := 0]\!] \left([\![\mathtt{P}_{assign}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right)\right\rangle\Big|_{c\neq 0} \\
&\quad + \left\langle (1-p)\,[\![c := 1]\!] \left([\![\mathtt{P}_{assign}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right)\right\rangle\Big|_{c\neq 0} \quad \text{(restriction linear (Theorem 3.4.2))} \\
&= (1-p)\,[\![c := 1]\!] \left([\![\mathtt{P}_{assign}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right)
\end{aligned}$$

With every loop execution $\vec{p}$ is permuted according to σ. So, for computing ${}_{\mathtt{P},G_\mu,B}n_1$ the probability on which the probabilistic decision is based on is $(B_\sigma \cdot \vec{p})_1$ and for ${}_{\mathtt{P},G_\mu,B}n_2$ on $(B_\sigma^2 \cdot \vec{p})_1$ and so on. Therefore, we have

$${}_{\mathtt{P},G_\mu,B}n_{k+1} = \left(1 - \left(B_\sigma^{k+1}\vec{p}\right)_1\right) [\![c := 1]\!] \left([\![\mathtt{P}_{assign}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right).$$

We prove that by induction over $k \in \mathbb{N}$. The proof can be found in the appendix (page **??**).

Since $\left(B_\sigma^k\vec{p}\right)_1 = p_{\sigma^k(1)}$ (Lemma 4.2.5), we have

$${}_{\mathtt{P},G_\mu,B}n_{k+1} = \left(1 - p_{\sigma^{k+1}(1)}\right) [\![c := 1]\!] \left([\![\mathtt{P}_{assign}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right).$$

We know that the assignment $[\![c := 1]\!]$ never changes the original value of c since c is 1 at the beginning and at the end of each non-terminating execution of the loop. Therefore, $[\![c := 1]\!]$ has no effect and can be omitted:

$${}_{\mathtt{P},G_\mu,B}n_{k+1} = \left(1 - p_{\sigma^{k+1}(1)}\right) \left([\![\mathtt{P}_{assign}]\!] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right).$$

We observe that $[\![\mathtt{P}_{prob}]\!]$ scales the PGF by a factor. Because $[\![\mathtt{P}_{assign}]\!]$ is linear (Theorem

3.4.2) we propose:

$$_{\mathtt{P},G_\mu,B}n_k = \left(\prod_{i=1}^{k}\left(1-p_{\sigma^i(1)}\right)\right)\left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right]^k\left({}_{\mathtt{P},G_\mu,B}n_0\right)\right) \qquad \text{for all } k\in\mathbb{N}. \tag{4.2}$$

The proof is done by induction and can be found in the appendix (page **??**).

However, we are not really interested how the PGF $_{\mathtt{P},G_\mu,B}n_0$ is modified in order to get the PGF $_{\mathtt{P},G_\mu,B}n_k$ for all $k\in\mathbb{N}$. We are interested in how a PGF $_{\mathtt{P},G_\mu,B}n_k$ differs from the PGF $_{\mathtt{P},G_\mu,B}n_{k+t}$. In order to obtain $_{\mathtt{P},G_\mu,B}n_{k+t}$ from $_{\mathtt{P},G_\mu,B}n_k$ we have to apply t-times $[\![\mathtt{P}_{prob}]\!]$ and t-times $[\![\mathtt{P}_{assign}]\!]$. Therefore, we propose:

$$_{\mathtt{P},G_\mu,B}n_{k+t} = \left(\prod_{i=1}^{t}\left(1-p_{\sigma^i(1)}\right)\right)\left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right]^t\left({}_{\mathtt{P},G_\mu,B}n_k\right)\right)$$

The proof is done by induction and can be found in the appendix (page **??**). It uses equation (4.2). The reason why $\left(\prod_{i=1}^{t}\left(1-p_{\sigma^i(1)}\right)\right)$ has as bounds 1 and t is due to the fact that we look only at the cycle $c_{\sigma,(1)}$ to obtain the current probability.

Now, we have to look at what $[\![\mathtt{P}_{assign}]\!]^t$ means for a PGF. Every application of $[\![\mathtt{P}_{assign}]\!]$ corresponds to an application of the function $f\colon \vec{x}\mapsto A_\pi\vec{x}+\vec{d}$. We know that $\left(f^t\left(\vec{x}\right)\right)_j = x_j + \frac{t}{len\left(c_{\pi,(j)}\right)}cyclesum_\pi(j)$ because of Lemma 4.2.16. So after t executions of the loop $\frac{t}{len\left(c_{\pi,(j)}\right)}cyclesum_\pi(j)$ has been added to each variable x_j. So it means that

$$\begin{aligned}&[\![\mathtt{P}_{assign}]\!]^t\\ &= \left[\!\!\left[x_1 := x_1 + \frac{t}{len\left(c_{\pi,(1)}\right)}cyclesum_\pi(1), ..., x_n := x_n + \frac{t}{len\left(c_{\pi,(n)}\right)}cyclesum_\pi(n)\right]\!\!\right].\end{aligned}$$

The ordering of the assignments $x_j := x_j + \frac{t}{len\left(c_{\pi,(j)}\right)}cyclesum_\pi(j)$ has no impact on the outcome since the right hand sides of the assignments contain no other variables than x_j. Before, we saw in the remark and Example 4.2.17 that $[\![x_j := x_j + i]\!](G_\mu) = X_j^i G_\mu$. That means that $\left[\!\!\left[x_j := x_j + \frac{t}{len\left(c_{\pi,(j)}\right)}cyclesum_\pi(j)\right]\!\!\right]$ has the same effect on the input PGF as multiplying the input PGF with $X_j^{\frac{t}{len\left(c_{\pi,(j)}\right)}cyclesum_\pi(j)}$. So we can write

$$\begin{aligned}{}_{\mathtt{P},G_\mu,B}n_{k+t} &= \left(\prod_{i=1}^{t}\left(1-p_{\sigma^i(1)}\right)\right)\left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right]^t\left({}_{\mathtt{P},G_\mu,B}n_k\right)\right)\\ &= \left(\prod_{i=1}^{t}\left(1-p_{\sigma^i(1)}\right)\right)\prod_{i=1}^{n}X_i^{cyclesum_\pi(i)\frac{t}{len\left(c_{\pi,(i)}\right)}}\left({}_{\mathtt{P},G_\mu,B}n_k\right)\end{aligned}$$

$$= c\left({}_{\mathtt{P},G_\mu,B}n_k\right).$$

□

Since we know in what what ${}_{\mathtt{P},G_\mu,B}n_k$ and ${}_{\mathtt{P},G_\mu,B}n_{k+t}$ differ from each other, we can look at how ${}_{\mathtt{P},G_\mu,B}n_k$ and ${}_{\mathtt{P},G_\mu,B}n_{k\%t}$ differ from each other. Since t applications of the loop body change ${}_{\mathtt{P},G_\mu,B}n_k$ only by a factor, a multiple of t applications changes ${}_{\mathtt{P},G_\mu,B}n_k$ by the power of that factor. That means that ${}_{\mathtt{P},G_\mu,B}n_{k+l\cdot t} = c^l\,{}_{\mathtt{P},G_\mu,B}n_k$. Since all $k > t-1$ can be written as $k = l\cdot t + j$ for some $j \in \{0,...,t-1\}$, we find a form for all ${}_{\mathtt{P},G_\mu,B}n_k$ based on the first $t-1$ ${}_{\mathtt{P},G_\mu,B}n_k$ and c.

Lemma 4.2.19. *For the loop of Program 5, the following holds:*

$$ {}_{P,G_\mu,B}n_k = {}_{P,G_\mu,B}n_{k\%t}\cdot c^{\lfloor\frac{k}{t}\rfloor} \qquad \text{for all } k\in\mathbb{N}$$

with $c = \prod_{i=1}^{t}\left(1-\vec{p}_{\sigma^i(1)}\right)\cdot\prod_{i=1}^{n}X_i^{\frac{cyclesum_\pi(i)}{len\left(c_{\pi,(i)}\right)}}$ *and* $t = len\,(c_{\sigma,1})\prod_{i=1}^{m}\left(len\,(c_{\pi,i})\right)$ *where m is the number of cycles of π.*

The proof is done by induction and can be found in the appendix (page **??**).

With the form we found for ${}_{\mathtt{P},G_\mu,B}n_k$ we can now deduce the form for $\left\langle[\![\mathtt{P}]\!]\left({}_{\mathtt{P},G_\mu,B}n_k\right)\right\rangle\Big|_{\neg B}$.

Theorem 4.2.20. *For the loop of Program 5, the following holds:*

$$\left\langle[\![\mathtt{P}]\!]\left({}_{\mathtt{P},G_\mu,B}n_k\right)\right\rangle\Big|_{\neg B} = b_{k\%t}c^{\lfloor\frac{k}{t}\rfloor} \qquad \text{for all } k\in\mathbb{N}$$

with $c = \left(\prod_{i=1}^{t}\left(1-p_{\sigma^i(1)}\right)\right)\cdot\prod_{i=1}^{n}X_i^{\frac{cyclesum_\pi(i)}{len\left(c_{\pi,(i)}\right)}}$, $b_{k\%t} = \langle[\![\mathtt{P}]\!]\,n_{k\%t}\rangle|_{\neg B}$ *and* $t = len\,(c_{\sigma,1})\prod_{i=1}^{m}\left(len\,(c_{\pi,i})\right)$ *where m is the number of cycles of π.*

Proof.

$$\begin{aligned}&\left\langle[\![\mathtt{P}]\!]\left({}_{\mathtt{P},G_\mu,B}n_k\right)\right\rangle\Big|_{\neg B}\\ =&\left\langle[\![\mathtt{P}_{prob}]\!]\left([\![\mathtt{P}_{assign}]\!]\left({}_{\mathtt{P},G_\mu,B}n_k\right)\right)\right\rangle\Big|_{\neg B}\end{aligned}$$

We now have to determine the probability on which the probabilistic choice is based on. We saw in the proof of Lemma 4.2.19 that if we have $[\![\mathtt{P}]\!]\left({}_{\mathtt{P},G_\mu,B}n_k\right)$ this probability is $p_{\sigma^{k+1}(1)}$.

$$\begin{aligned}&\left\langle[\![\mathtt{P}]\!]\left({}_{\mathtt{P},G_\mu,B}n_k\right)\right\rangle\Big|_{\neg B}\\ =&\left\langle[\![\mathtt{P}_{prob}]\!]\left([\![\mathtt{P}_{assign}]\!]\left({}_{\mathtt{P},G_\mu,B}n_k\right)\right)\right\rangle\Big|_{\neg B}\end{aligned}$$

$$
\begin{aligned}
&= \left\langle p_{\sigma^{k+1}(1)} \left[\!\left[c := 0\right]\!\right] \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right) + \left(1 - p_{\sigma^{k+1}(1)}\right) \left[\!\left[c := 1\right]\!\right] \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right)\right\rangle\Big|_{\neg B} \\
&= p_{\sigma^{k+1}(1)} \left[\!\left[c := 0\right]\!\right] \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right) \\
&= p_{\sigma^{k+1}(1)} \left[\!\left[c := 0\right]\!\right] \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right] \left(c^{\lfloor \frac{k}{t} \rfloor} {}_{\mathtt{P},G_\mu,B}n_{k\%t}\right)\right) && \text{(Lemma 4.2.19)} \\
&= p_{\sigma^{k+1}(1)} \left[\!\left[c := 0\right]\!\right] \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right] \left(\left(\left(\prod_{i=1}^{t} \left(1 - p_{\sigma^i(1)}\right)\right) \cdot \prod_{i=1}^{n} X_i^{cyclesum_\pi(i)\frac{t}{len\left(c_{\pi,(i)}\right)}}\right)^{\lfloor \frac{k}{t} \rfloor} {}_{\mathtt{P},G_\mu,B}n_{k\%t}\right)\right) \\
& && \text{(definition of } c\text{)} \\
&= \left(\prod_{i=1}^{t} \left(1 - p_{\sigma^i(1)}\right)\right)^{\lfloor \frac{k}{t} \rfloor} p_{\sigma^{k+1}(1)} \left[\!\left[c := 0\right]\!\right] \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right] \left(\left(\prod_{i=1}^{n} X_i^{cyclesum_\pi(i)\frac{t}{len\left(c_{\pi,(i)}\right)}}\right)^{\lfloor \frac{k}{t} \rfloor} {}_{\mathtt{P},G_\mu,B}n_{k\%t}\right)\right) \\
& && \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right] \text{ linear}\right)
\end{aligned}
$$

We saw in the proof of Lemma 4.2.19 that $\prod_{i=1}^{n} X_i^{cyclesum_\pi(i)\frac{t}{len\left(c_{\pi,(i)}\right)}}$ has the same effect on a PGF as $\left[\!\left[\mathtt{P}_{assign}\right]\!\right]^t$.

$$
\begin{aligned}
&\left\langle \left[\!\left[\mathtt{P}\right]\!\right] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right\rangle\Big|_{\neg B} \\
&= \left(\prod_{i=1}^{t} \left(1 - p_{\sigma^i(1)}\right)\right)^{\lfloor \frac{k}{t} \rfloor} p_{\sigma^{k+1}(1)} \left[\!\left[c := 0\right]\!\right] \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right] \left(\left(\prod_{i=1}^{n} X_i^{cyclesum_\pi(i)\frac{t}{len\left(c_{\pi,(i)}\right)}}\right)^{\lfloor \frac{k}{t} \rfloor} {}_{\mathtt{P},G_\mu,B}n_{k\%t}\right)\right) \\
&= \left(\prod_{i=1}^{t} \left(1 - p_{\sigma^i(1)}\right)\right)^{\lfloor \frac{k}{t} \rfloor} p_{\sigma^{k+1}(1)} \left[\!\left[c := 0\right]\!\right] \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right] \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right]^{\lfloor \frac{k}{t} \rfloor \cdot t} {}_{\mathtt{P},G_\mu,B}n_{k\%t}\right)\right) \\
&= \left(\prod_{i=1}^{t} \left(1 - p_{\sigma^i(1)}\right)\right)^{\lfloor \frac{k}{t} \rfloor} p_{\sigma^{k+1}(1)} \left[\!\left[c := 0\right]\!\right] \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right]^{\lfloor \frac{k}{t} \rfloor \cdot t} \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right] {}_{\mathtt{P},G_\mu,B}n_{k\%t}\right)\right)
\end{aligned}
$$

Since $\left[\!\left[\mathtt{P}_{assign}\right]\!\right]$ has no effect on the value of c we can change the order of applications of $\left[\!\left[\mathtt{P}_{assign}\right]\!\right]$ and $\left[\!\left[c := 0\right]\!\right]$.

$$
\begin{aligned}
&\left\langle \left[\!\left[\mathtt{P}\right]\!\right] \left({}_{\mathtt{P},G_\mu,B}n_k\right)\right\rangle\Big|_{\neg B} \\
&= \left(\prod_{i=1}^{t} \left(1 - p_{\sigma^i(1)}\right)\right)^{\lfloor \frac{k}{t} \rfloor} p_{\sigma^{k+1}(1)} \left[\!\left[c := 0\right]\!\right] \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right]^{\lfloor \frac{k}{t} \rfloor \cdot t} \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right] {}_{\mathtt{P},G_\mu,B}n_{k\%t}\right)\right) \\
&= \left(\prod_{i=1}^{t} \left(1 - p_{\sigma^i(1)}\right)\right)^{\lfloor \frac{k}{t} \rfloor} p_{\sigma^{k+1}(1)} \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right]^{\lfloor \frac{k}{t} \rfloor \cdot t} \left(\left[\!\left[c := 0\right]\!\right] \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right] {}_{\mathtt{P},G_\mu,B}n_{k\%t}\right)\right)\right) \\
&= \left(\prod_{i=1}^{t} \left(1 - p_{\sigma^i(1)}\right)\right)^{\lfloor \frac{k}{t} \rfloor} \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right]^{\lfloor \frac{k}{t} \rfloor \cdot t} \left(p_{\sigma^{k+1}(1)} \left[\!\left[c := 0\right]\!\right] \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right] {}_{\mathtt{P},G_\mu,B}n_{k\%t}\right)\right)\right) \\
& && \left(\left[\!\left[\mathtt{P}_{assign}\right]\!\right] \text{ linear}\right)
\end{aligned}
$$

We know that $k = \left\lfloor \frac{k}{t} \right\rfloor \cdot t + k\%t$. In addition, t is a multiple of $len\left(c_{\sigma,(1)}\right)$. With Lemma 4.2.9 we can therefore deduce

$$\sigma^{k+1}(1) = \sigma^{\left\lfloor \frac{k}{t} \right\rfloor \cdot t + k\%t+1}(1) = \sigma^{k\%t+1}(1)$$

With that there follows for $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B} n_k \right) \right\rangle \Big|_{\neg B}$:

$$\begin{aligned}
&\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B} n_k \right) \right\rangle \Big|_{\neg B} \\
=&\left(\prod_{i=1}^{t} \left(1 - p_{\sigma^i(1)}\right) \right)^{\left\lfloor \frac{k}{t} \right\rfloor} \left([\![\mathtt{P}_{assign}]\!]^{\left\lfloor \frac{k}{t} \right\rfloor \cdot t} \left(p_{\sigma^{k+1}(1)} \, [\![c := 0]\!] \, [\![\mathtt{P}_{assign}]\!] \, n_{k\%t} \right) \right) \\
=&\left(\prod_{i=1}^{t} \left(1 - p_{\sigma^i(1)}\right) \right)^{\left\lfloor \frac{k}{t} \right\rfloor} \left([\![\mathtt{P}_{assign}]\!]^{\left\lfloor \frac{k}{t} \right\rfloor \cdot t} \left(p_{\sigma^{k\%t+1}(1)} \, [\![c := 0]\!] \, [\![\mathtt{P}_{assign}]\!] \, n_{k\%t} \right) \right) \\
=&\left(\prod_{i=1}^{t} \left(1 - p_{\sigma^i(1)}\right) \right)^{\left\lfloor \frac{k}{t} \right\rfloor} \left([\![\mathtt{P}_{assign}]\!]^{\left\lfloor \frac{k}{t} \right\rfloor \cdot t} \, \langle [\![\mathtt{P}]\!] \, n_{k\%t} \rangle |_{\neg B} \right)
\end{aligned}$$

We saw that the effect of $[\![\mathtt{P}_{assign}]\!]^t$ on a PGF is the same as multiplying it with $\prod_{i=1}^{n} X_i^{cyclesum_\pi(i) \frac{t}{len\left(c_{\pi,(i)}\right)}}$. Therefore, we have

$$\begin{aligned}
&\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B} n_k \right) \right\rangle \Big|_{\neg B} \\
=&\left(\prod_{i=1}^{t} \left(1 - p_{\sigma^i(1)}\right) \right)^{\left\lfloor \frac{k}{t} \right\rfloor} \left\langle [\![\mathtt{P}_{assign}]\!]^{\left\lfloor \frac{k}{t} \right\rfloor \cdot t} \left([\![\mathtt{P}]\!] \, n_{k\%t} \right) \right\rangle \Big|_{\neg B} \\
=&\left(\prod_{i=1}^{t} \left(1 - p_{\sigma^i(1)}\right) \right)^{\left\lfloor \frac{k}{t} \right\rfloor} \left(\prod_{i=1}^{n} X_i^{cyclesum_\pi(i) \frac{t}{len\left(c_{\pi,(i)}\right)}} \right)^{\left\lfloor \frac{k}{t} \right\rfloor} \langle [\![\mathtt{P}]\!] \, n_{k\%t} \rangle |_{\neg B} \\
=&c^{\left\lfloor \frac{k}{t} \right\rfloor} b_{k\%t}
\end{aligned}$$

□

The form we could now prove for $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B} n_k \right) \right\rangle \Big|_{\neg B}$ is exactly the form we need for the second proof rule, Theorem 4.1.10. So for all programs of the form of Program 5 we can find a closed form according to the form of $\left\langle [\![\mathtt{P}]\!] \left({}_{\mathtt{P},G_\mu,B} n_k \right) \right\rangle \Big|_{\neg B}$ (Theorem 4.2.20).

Corollary 4.2.21. *Because of Theorem 4.2.20 and Theorem 4.1.10 we can find for* $[\![\mathtt{while}(B)\{\mathtt{P}\}]\!](G_\mu)$ *with a while-loop as in Program 5 a closed form. This closed form*

is given by

$$[\![\texttt{while}(B)\{P\}]\!](G_\mu) = \sum_{j=0}^{t-1} \frac{b_j}{1-c_j} + {}_{P,G_\mu,B}e_0 \quad .$$

for $t, b_0, ..., b_{t-1}, c_0, ..., c_{t-1}$ *from Theorem 4.2.20*

Chapter 5

Conclusion

In this bachelor thesis, we developed rules for a semantics based on PGFs. We proved that those rules are well-defined and linear. This is a good starting point for future use of this semantics. Furthermore, we introduced two proof rules to deal with while-loops. Those rules simplify the computation of the result of a while-loop. We presented one form of a while-loop these rules can always be applied to. This form allows the while-loop to contain a probabilistic decision which is based on a set of different probabilities. Furthermore, in the loop body, a set of variables can be summed up with a constant and be permuted. That way we can apply the proof rule to some often-used examples for probabilistic programs like the duelling cowboys program.

Further ideas would be to look at other forms of while-loops and whether they can be analysed with the existing proof rules. In addition, the general question – i.e., in which cases a closed form for the PGF resulting from a while-loop exists or not – might be interesting to discuss. This is due to the fact that it is easier to extract stochastic measurements like expected values from PGFs with a closed form.

This bachelor thesis mainly dealt with while-loops where the variables are only modified by addition with constants. Another idea would be to think about while-loops where variables are multiplied with constants or with another. Not only the modification of the variables could be different but also the modification of the probability. The form of program we looked at has a fixed set of probabilities the probabilistic decision is based on. However, we could construct a program where this probability changes over the course of the execution. Additionally, we could look at loops with more than one probabilistic decision. We might even be able to analyse some of these loops with the existing proof rules. For some others, we might come up with new proof rules.

The PGF resulting from a while-loop always had a closed form for all examples and

the form of while-loop we looked at. It would be interesting to think about kinds of while-loops for which such a closed form exists or in what cases we might even be able to exclude that there is such a closed form.

Chapter 6

Appendix

Bibliography

[AGM94] S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors. *Handbook of Logic in Computer Science: Volume 3. Semantic Structures*. Handbook of Logic in Computer Science. Clarendon Press, 1994.

[Bos13] S. Bosch. *Algebra*. Springer-Verlag Berlin Heidelberg, 8th edition, 2013.

[CK08] E. Cramer and U. Kamps. *Grundlagen der Wahrscheinlichkeitsrechnung und Statistik*. Springer-Verlag Berlin Heidelberg, 2nd edition, 2008.

[Den03] K. Denecke. *Algebra und Diskrete Mathematik für Informatiker*. Teubner Verlag, 2003.

[DR08] W. Dahmen and A. Reusken. *Numerik Für Ingenieure und Naturwissenschaftler*. Springer-Verlag Berlin Heidelberg, 2nd edition, 2008.

[Fis10] G. Fischer. *Lineare Algebra: Eine Einführung für Studienanfänger*. Vieweg+Teubner Verlag, 17th edition, 2010.

[For11] O. Forster. *Analysis 1*. Vieweg+Teubner Verlag, 10th edition, 2011.

[Gil87] J.R. Giles. *Introduction to the Analysis of Metric Spaces*. Cambridge University Press, 1987.

[GKP89] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley Longman Publishing Co., 1989.

[Gor79] J.C. Gordon. *The Denotational Description of Programming Languages: An Introduction*. Springer-Verlag New York, 1979.

[Grö49a] W. Gröbner. *Moderne Algebraische Geometrie: Die Idealtheoretischen Grundlagen*. Springer-Verlag Wien, 1949.

[Grö49b] W. Gröbner. *Moderne Algebraische Geometrie: Die Idealtheoretischen Grundlagen*. Springer Vienna, 1949.

[Heu06] H. Heuser. *Lehrbuch der Analysis.* Vieweg+Teubner Verlag, 16th edition, 2006.

[Jän08] K. Jänich. *Lineare Algebra.* Springer-Verlag Berlin Heidelberg, 11th edition, 2008.

[JKB97] N.L. Johnson, S. Kotz, and N. Balakrishnan. *Discrete Multivariate Distributions.* Wiley, 1997.

[JKK92] N.L. Johnson, A.W. Kemp, and S. Kotz. *Univariate Discrete Distributions.* Wiley, 2nd edition, 1992.

[KM10] C. Karpfinger and K. Meyberg. *Algebra Gruppen - Ringe - Körper.* Spektrum Akademischer Verlag, 2nd edition, 2010.

[LSS84] J. Loeckx, K. Sieber, and R.D. Stansifer. *The foundations of program verification.* Teubner, Wiley, 1984.

[Rud76] W. Rudin. *Principles of Mathematical Analysis.* 3rd edition, 1976.

[SP08] Rainer Schulze-Pillot. *Einführung in Algebra und Zahlentheorie.* Springer-Verlag Berlin Heidelberg, 2nd edition, 2008.

[Sut09] W.A. Sutherland. *Introduction to Metric and Topological Spaces.* Oxford University Press, 2nd edition, 2009.

[Tri02] Kishor S. Trivedi. *Probability and Statistics with Reliability, Queuing and Computer Science Applications.* Wiley, 2nd edition, 2002.

[Wil90] Herbert S. Wilf. *Generatingfunctionology.* Academic Press, 1990.

[Win93] G. Winskel. *The Formal Semantics of Programming Languages: An Introduction.* MIT Press, 1993.