RWTH AACHEN UNIVERSITY

Extending Probability Generating Function Semantics to Negative Variable Valuations

Bachelor's Thesis

of Simon Feiden

March 29, 2016

Advisor
Benjamin Kaminski

First Reviewer
Prof. Dr. Ir. Joost-Pieter Katoen

Second Reviewer
apl. Prof. Dr. Thomas Noll

Eidesstattliche Versicherung

Name, Vorname

Matrikelnummer (freiwillige Angabe)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum

Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Ort, Datum

Unterschrift

Abstract

We present a semantics for a probabilistic programming language with program variables that can have values in the integers. The semantics is based on formal power series and is an extension to previous work of Scherbaum [**?**]. Scherbaum's semantics does not allow program variables to have negative valuations. We define a semantic rule for each of the operations in the programming language. Loops are the most intricate operations. To ease finding a loop's semantics, we introduce a novel concept to obtain a statement about the semantics of a loop. It uses binary relations to either find the actual loop semantics or prove an overapproximation of the loop semantics. An element of such a relation is a pair of input and output of a loop. Later on, we prove that our semantics is equivalent in expectation to an already established one. For the comparison, we chose the weakest preexpectation semantics introduced by McIver and Morgan. We prove that the execution of a program in either semantics gives the same expected value for an arbitrary property expressed as a function over the program variables.

Contents

1 Introduction

Probabilistic programs are programs that do not always produce the same result for one set of inputs. This is achieved by the usage of probabilistic operations, which are embedded into the programming language itself. Applications are in testing, where error rates can be modelled easily using probabilistic programs. They can also be used to solve problems that are very hard to solve or are not solvable at all using classical programming paradigms [**?**].
Because of its non-deterministic nature, code written by using a probabilistic programming language is not easily verified just by reading and understanding. To obtain insights about properties such as correctness and termination probability, a formal method is necessary. It should allow to derive statements about the aforementioned properties. Unlike program code, which describes how something is calculated, the formal method must keep track of what happens while a program is executed. What happens, or the meaning of the process, is called the program's semantics. One idea to define a semantics is to keep track of all possible variable valuations, called the program states.
Probabilistic programs are not deterministic. Consequently, program states only have a probability of appearing as result of the whole computation. To capture all program states with their probabilities, *probability generating functions* (PGFs) can be used. This thesis is built on work of Scherbaum [**?**], which uses *formal power series* as PGFs. These PGFs lack the possibility of having variables with negative values. Thus, even simple programs that use subtraction cannot be represented by one of those PGFs. In this work, we will extend Scherbaum's model, so that variables can also have negative values. Later, we will show that the newly derived semantics is equivalent to an already established one which was introduced by McIver and Morgan. [**?**].

2 Preliminaries

2.1 Formal Power Series

A formal power series (FPS) [**?**] is a polynomial of degree ∞ over one variable, most commonly X. A formal power series F is an object of the form

$$F = a_0X^0 + a_1 \cdot X^1 + a_2X^2 + \ldots = \sum_{i=0}^{\infty} a_iX^i$$

The coefficients a_i are real numbers. When dealing with a formal power series F, F is not meant to be a function. This means that X is not necessarily substituted by a value, so properties like convergence are ignored.
Formal power series can be added and multiplied. Addition comes in a very natural way, it is done componentwise:

$$F + G = \sum_{i=0}^{\infty} a_iX^i + \sum_{i=0}^{\infty} b_iX^i = \sum_{i=0}^{\infty}(a_i + b_i)X^i$$

Multiplication is more complicated.

$$F \cdot G = \sum_{i=0}^{\infty} c_iX^i, \text{ where } c_i = \sum_{k=0}^{i} a_kb_{i-k}$$

The product of two formal power series is always defined because the sum of every coefficient c_i consists of only finitely many elements. In fact, the formal power series form a ring under addition and multiplication. The ring's additive and multiplicative identities are

$$0 = \sum_{i=0}^{\infty} 0X^i \text{ and } 1 = 1X^0 + 0X^1 + 0X^2 + \ldots$$

Later on, we will need another operation on formal power series, the scalar multiplication:

$$\alpha \cdot F = \sum_{i=0}^{\infty}(\alpha \cdot a_i)X^i, \text{ where } \alpha \in \mathbb{R}$$

Obviously, scalar multiplication always has a formal power series as its result. Some formal power series even have a closed form. A closed form is desirable because it eases readability and calculation. As an example, let us have a look at the geometric series. It comes up often and is defined as follows:

$$G = 1X^0 + pX^1 + p^2X^2 + \ldots = \sum_{i=0}^{\infty} p^iX^i, \text{ where } |p| < 1$$

The closed form of this series is

$$G = \frac{1}{1 - pX}$$

This means that $1 - pX$ is the multiplicative inverse of G. In other words: $G \cdot (1 - pX) = 1$ where 1 on the right hand side is the identity of the ring, as given above. In fact, the equation also holds if $|p| > 1$ because we do not consider convergence of the sums. $|p| < 1$ must only hold if we want to call a PGF a geometric series.

Absolute Value

Despite ignoring convergence etc. during operations on formal power series, it is sometimes useful to know about the sum of all coefficients. Summing all coefficients is comparable to substituting the formal variable X with 1, or evaluating the series at 1. It may happen that the sum diverges, but when it does not, it can have a meaningful interpretation as we will see later. Given a formal power series $G = \sum_{i=0}^{\infty} a_i X^i$, define the absolute value $|G|$ as the sum of all its coefficients:

$$|G| = \sum_{i=0}^{\infty} a_i$$

Here, we have another reason why closed forms of formal power series are interesting. Consider the geometric series

$$G = \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i X^i$$

Now, calculate the absolute value of both the closed and non closed form.

$$|G| = \left| \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i X^i \right| = \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = 2$$

$$|G| = \left| \frac{1}{1 - \frac{1}{2}X} \right| = \frac{1}{1 - \frac{1}{2}} = 2$$

Of course, they both have the same value because they are equivalent. Nonetheless, calculating the absolute value of the closed form is easier, since fractional arithmetic is simpler than reasoning about the value of a series.
Note that, although the notation may suggest it, the absolute value is *not* a norm in the classical sense. It is rather a linear function because the following holds:

$$|a \cdot G + F| = a \cdot |G| + |F|$$

2.2 Multivariate Formal Power Series

Multivariate formal power series are similar to formal power series except they can have more than one formal variable. A multivariate formal power series G is an object of the form

$$G = \sum_{s \in \mathbb{N}^k} a_s X_1^{s_1} \cdots X_k^{s_k}, \text{ where } k \in \mathbb{N}_{>0}$$

Addition is defined componentwise as it is for formal power series.

$$F + G = \sum_{s \in \mathbb{N}^k} a_s X_1^{s_1} \cdots X_k^{s_k} + \sum_{s \in \mathbb{N}^k} b_s X_1^{s_1} \cdots X_k^{s_k} = \sum_{s \in \mathbb{N}^k} (a_s + b_s) X_1^{s_1} \cdots X_k^{s_k}$$

Multiplication again is more complicated, but is defined in a similar fashion as for formal power series.

$$F \cdot G = \sum_{s \in \mathbb{N}^k} c_s X_1^{s_1} \cdots X_k^{s_k}$$

$$\text{where } c_s = \sum_{\substack{n \in \mathbb{N}^k \wedge \\ n_j \leq s_j \text{ for all } j \in \{1,\ldots,k\}}} a_{n_1,\ldots,n_k} \cdot b_{s_1-n_1,\ldots,s_k-n_k}$$

The concepts of scalar multiplication and absolute value are defined as expected for multivariate formal power series.

2.3 Formal Power Series As Probability Generating Functions

We mentioned earlier that formal power series can be used as probability generating functions. Given a formal power series

$$G = \sum_{s \in \mathbb{N}^k} \mu_s X_1^{s_1} \cdots X_k^{s_k}$$

μ_s is interpreted as the probability that the random variables $X_1, \ldots, X_k$ have the values $s_1, \ldots, s_k$. In other words

$$\mathcal{P}(X_1 = s_1, \ldots, X_k = s_k) = \mu_s$$

When used as PGFs, formal power series can still be added, scaled and multiplied.

2.4 Programs

In order to reason about probabilistic programs by means of a semantics, we need to define a programming language. We will use the same language as in [**?**]. It consists of seven different operations that can be combined to form programs.

Basic Operations

1. *skip*

   The *skip* operation does nothing and simply goes on to the next operation. It is useful in composite operations.

2. *abort*

   The *abort* operation does not terminate.

3. $X := e$

   The variable X is assigned the value of e. e is an expression over the program variables that is evaluated right before the assignment to X.

Composite Operations

Here, P, P_1 and P_2 are programs and B is a Boolean condition.

1. $P_1; P_2$

   The concatenation $P_1; P_2$ means to first execute P_1 and then P_2.

2. $\mathit{if}(B)\,\{\,P_1\,\}\,\mathit{else}\,\{\,P_2\,\}$

   If the current variable valuations satisfy the condition B then P_1 is executed, otherwise P_2 is executed.

3. $\{P_1\}[p]\{P_2\}$ for some $p \in [0, 1]$

   This statement is called probabilistic choice and is what distinguishes this programming language from classical ones. Either P_1 or P_2 is executed at random. P_1 is executed with probability p and P_2 is executed with probability $1 - p$.

4. $\mathit{while}\,(B)\{P\}$

   The *loop body* P is repeatedly executed until the *loop condition* B is no longer satisfied by the program variables.

2.5 PGF Semantics

In [**?**], a semantics for probabilistic programs with multivariate power series as probability generating functions is introduced. In the following, let P be a program and G be a PGF with

$$G = \sum_{s \in \mathbb{N}^k} \mu_s X_1^{s_1} \cdots X_k^{s_k}$$

where k is equal to the number of program variables used in P.
Every element of the sum in G corresponds to one possible program state with its probability of occurrence. For example

$$\frac{1}{3} X_1^2 X_2^5$$

means that with probability $\frac{1}{3}$, variable X_1 has value 2 and variable X_2 has value 5. It is easy to see that the absolute value $|G|$ must be less than or equal to 1, because otherwise the total probability of all program states would be greater than 1. Note that the sum of a PGF ranges over $\mathbb{N}^k$, so a variable cannot have a negative value. Assignments or rather expressions are constrained to only evaluate to positive values.
Every P has a corresponding function, denoted by $[\![P]\!]$, that transforms an initial PGF into the PGF that results from executing P. These functions are defined next.

Basic Operations

1. $[\![\mathit{skip}]\!]\,(G) = G$
2. $[\![\mathit{abort}]\!]\,(G) = 0 = \sum_{s \in \mathbb{N}^k} 0 X_1^{s_1} \cdots X_k^{s_k}$
3. $[\![X_j := e]\!]\,(G) = \sum_{s \in \mathbb{N}^k} \mu_s X_1^{s_1} \cdots X_j^{e(s)} \cdots X_k^{s_k}$

4. $\langle G \rangle_B = \sum_{s \in \mathbb{N}^k} \begin{cases} \mu_s X_1^{s_1} \cdots X_k^{s_k} & \text{if } s \models B \\ 0 & \text{otherwise} \end{cases}$

Above, s is an element of $\mathbb{N}^k$. s_j refers to the jth components of s. Of course, $1 \leq j \leq k$.

Composite Operations

Here, P, P_1, and P_2 are programs and B is a Boolean condition.

1. $[\![P_1; P_2]\!](G) = [\![P_2]\!]([\![P_1]\!](G))$
2. $[\![\{P_1\}[p]\{P_2\}]\!](G) = p \cdot [\![P_1]\!](G) + (1-p) \cdot [\![P_2]\!](G)$
3. $[\![\mathit{if}(B)\,\{\,P_1\,\}\,\mathit{else}\,\{\,P_2\,\}]\!](G) = [\![P_1]\!](\langle G \rangle_B) + [\![P_2]\!](\langle G \rangle_{\neg B})$
4. $[\![\mathit{while}\,(B)\{P\}]\!](G) = \sup_{n \in \mathbb{N}} \left\{ \left\langle \mathrm{H}^n_{P,B}(G) \right\rangle_{\neg B} \right\}$, where $\mathrm{H}_{P,B}(G) = [\![P]\!](\langle G \rangle_B) + \langle G \rangle_{\neg B}$
   $\mathrm{H}_{P,B}$ represents a single iteration of the loop body.

Similar to how programs are concatenated to form longer programs, their individual semantics functions are composed to form the semantics function of the longer program.
If G' results from executing a program P on an initial PGF G, i.e. $G' = [\![P]\!](G)$, then $|G'|$ is the termination probability of P given the initial distribution. A simple example is *abort*. Applying *abort* to any PGF always results in 0, which has absolute value 0, meaning its termination probability is 0. This is exactly how *abort* is defined. Later, we will see loops that have a termination probability of less than 1.

Listing 1: Canonical example

```
X := 0;
F := 0;
while ( F = 0 ) {
  { X := X + 1 }[0.5]{ F := 1 }
}
```

Listing 2: Inverse canonical example

```
X := 0;
F := 0;
while ( F = 0 ) {
  { X := X - 1 }[0.5]{ F := 1 }
}
```

Figure 1: Two simple probabilistic programs.

3 Extended Semantics

Having introduced a probabilistic programming language and an associated semantics, we now want to extend that semantics so that variables can also have negative valuations. But first, let us have a look at why this is beneficial. Even simple programs like $X := -1$ are not representable in the previous semantics because X is assigned a negative value. This might be a contrived example but for instance, every subtraction that is used in a program must be checked by hand to never have a negative result. Consider Listings **??** and **??** in Figure **??**. The first one is perfectly expressible in the semantics while the latter is not, although only a single operation was changed from + to −.
We will improve on this by defining a new semantics that allows negative variable evaluations. Additionally, the new semantics should also support closed forms. The first is to simply extend the range of the PGFs from $\mathbb{N}^k$ to $\mathbb{Z}^k$, i.e.

$$G = \sum_{s \in \mathbb{Z}^k} s X_1^{s_1} \cdots X_k^{s_k}$$

Those are still PGFs, but they no longer form a ring, because multiplication is not well-defined for every pair of PGFs with extended range. Unfortunately, a ring is necessary for closed forms to be well-defined, so we cannot use PGFs with extended range for the new semantics. A solution to the problem is as follows: Given k program variables, the set $\mathbb{S}$ of all possible variable evaluations, called *state space*, is $\mathbb{S} = \mathbb{Z}^k$. We will partition $\mathbb{S}$ into sets $S_1, \ldots, S_{2^k}$ such that every program variable X_j always has the same sign in every state in S_j. For example, if a program has 1 variable, so $k = 1$, then the state space $\mathbb{S} = \mathbb{Z}$. The two partitions are

$$S_1 = \mathbb{N}$$
$$S_2 = \mathbb{Z} \setminus \mathbb{N} = \{-1, -2, \ldots\} =: -\mathbb{N}^*$$

If a program has 2 variables, $\mathbb{S} = \mathbb{Z}^2$ and the 4 partitions are

$$S_1 = \mathbb{N} \times \mathbb{N}$$
$$S_2 = -\mathbb{N}^* \times \mathbb{N}$$
$$S_3 = \mathbb{N} \times -\mathbb{N}^*$$
$$S_4 = -\mathbb{N}^* \times -\mathbb{N}^*$$

In the general case with $k > 0$ variables,

$$S_j = \bigtimes_{i=0}^{k} \begin{cases} \mathbb{N} & \text{if } b(j)_i = 0 \\ -\mathbb{N}^* & \text{otherwise} \end{cases}, \text{ where } 1 \leq j \leq 2^k$$

where $\bigtimes$ is the generalised cartesian product and $b(j)$ is the j-th element for some enumeration $b\colon \{1, \ldots, 2^k\} \to \{0,1\}^k$. As $b(j)$ is an element of a cartesian product, it is a tuple. Thus, $b(j)_i$ is simply the i-th entry of $b(j)$. $\{0,1\}^k$ has 2^k elements, just as many as there are partitions. The ith entry of a tuple in $\{0,1\}^k$ determines if the program variable X_i is negative or positive in the corresponding partition.
Having partitioned the state space, we can now introduce the new semantics. It consists of tuples which have an entry for every partition of the state space. Every entry of such a tuple is a multivariate formal power series with its range being the corresponding partition.

Definition 3.1 (Semantic Tuple)**.** *Let P be a program of k variables. A* semantic tuple T_G *is an object of the form*

$$T_G = \left(\sum_{s \in S_1} \mu_s X_1^{s_1} \cdots X_k^{s_k}, \ldots, \sum_{s \in S_{2^k}} \mu_s X_1^{s_1} \cdots X_k^{s_k} \right)$$

Because we are using multivariate formal series which have closed forms, we can use them in the entries of the tuples. Let us look at an example using the following program P.

```
X := 0;
{ X := -9 }[0.6]{ X := 9 }
```

P has one variable, so the semantic tuples have two entries. The first entry contains the part where X is positive and the second entry the part where X is strictly negative. After executing the program, X has value -9 with probability 0.6 and value 9 with probability 0.4. A corresponding semantic tuple T_G in the semantics is

$$T_G = (0.4X^9, 0.6X^{-9})$$

Unfortunately, the semantics becomes unwieldy when dealing with many variables, because the number of entries grows exponentially in the number of variables. With three variables the tuples have eight entries and with four variables they have sixteen entries. Handling sixteen or more entries is not convenient when calculating a semantics by hand. To avoid this problem, we can fall back to using formal power series with an extended range.

Theorem 3.1. *Every PGF with extended range*

$$G = \sum_{s \in \mathbb{Z}^k} \mu_s X_1^{s_1} \cdots X_k^{s_k}$$

corresponds to the semantic tuple

$$T_G = \left(\sum_{s \in S_1} \mu_s X_1^{s_1} \cdots X_k^{s_k}, \ldots, \sum_{s \in S_{2^k}} \mu_s X_1^{s_1} \cdots X_k^{s_k} \right)$$

Every G is easily decomposable into entries of a semantic tuples. Furthermore, the extended FPSs can be added and scaled, just as the normal PGFs.

Theorem 3.2. *The set of FPSs with extended range combined with addition and scalar multiplication forms a* vector space.

Proof. See Appendix **??**. □

The semantics only needs these two operations to be defined, so we do not lack any expressive power from using them. In consequence, we can use them to simplify the notation of the semantics when dealing with many variables. Additionally, the entries of the semantics tuples are not always non-zero. Consider the examples in Figure **??**. All of them contain a variable F that is only ever assigned the value 0 or 1, so F is always positive. The entries where F is negative are therefore 0, so they do not have to be considered when calculating a semantics.

3.1 Programs without Loops

Having defined objects that can model the whole state space, we can now go on to define what happens to them during the execution of a program. First, we will treat programs without loops because their semantics is straightforward while the semantics for loops requires some more work. We will adopt the notation that $[\![P]\!]$ is the semantics function corresponding to P.

Basic operations

1. $[\![skip]\!](G) = G$
2. $[\![abort]\!](G) = 0 = \sum_{s \in \mathbb{Z}^k} 0 X_1^{s_1} \cdots X_k^{s_k}$
3. $[\![X_j := e]\!](G) = \sum_{s \in \mathbb{Z}^k} \mu_s X_1^{s_1} \cdots X_j^{e(s)} \cdots X_k^{s_k}$

Composite operations

Here, P, P_1 and P_2 are programs and B is a Boolean condition.

1. $[\![P_1; P_2]\!](G) = [\![P_2]\!]([\![P_1]\!](G))$
2. $[\![\{P_1\}[p]\{P_2\}]\!](G) = p \cdot [\![P_1]\!](G) + (1-p) \cdot [\![P_2]\!](G)$
   Note that both operations of the vector space are used.
3. $\left\langle \sum_{s \in \mathbb{Z}^k} \mu_s X_1^{s_1} \cdots X_k^{s_k} \right\rangle_B = \sum_{s \in \mathbb{Z}^k} \begin{cases} \mu_s X_1^{s_1} \cdots X_k^{s_k} & \text{if } s \models B \\ 0 & \text{otherwise} \end{cases}$
4. $[\![if(B)\,\{\,P_1\,\}\,else\,\{\,P_2\,\}]\!](G) = [\![P_1]\!](\langle G \rangle_B) + [\![P_2]\!](\langle G \rangle_{\neg B})$

3.2 Domains

We now want to define the semantics of a loop $\mathit{while}(B)\{P\}$. As has been done before in [**?**] and [**?**], we will treat the loop semantics as the least fixed point of a function on an *ω-complete partial order*, also called *domain*. For the loop semantics, we need two ω-complete partial orders, one containing all PGFs and the other all PGF-transformers.
An ω-complete partial order [**?**] is a set equipped with a relation $\sqsubseteq$ which is a *partial order* and has a completeness property: $(P, \sqsubseteq)$ is a partial order if and only if $\sqsubseteq$ is reflexive, antisymmetric and transitive. $(P, \sqsubseteq)$ is ω-complete if every infinitely ascending chain $d_1 \sqsubseteq d_2 \sqsubseteq \ldots \in P$ has a supremum.

Definition 3.2. *The set D_k is the set of all PGFs of k variables whose coefficients and absolute value lie in the range $[0, 1]$.*

$$D_k = \left\{ \sum_{s \in \mathbb{Z}^k} \mu_s X_1^{s_1} \cdots X_k^{s_k} \middle| \forall s \in \mathbb{Z}^k \colon \mu_s \in [0,1] \wedge \sum_{s \in \mathbb{Z}^k} \mu_s \in [0,1] \right\}$$

The set L_k is the set of all functions from D_k to D_k.

$$L_k = \{D_k \to D_k\}$$

We equip the sets D_k and L_k with the following relations:

- $d \sqsubseteq_{D_k} d' \iff \forall s \in \mathbb{Z}^k \colon \mu_s \leq \mu'_s$
  A PGF d is less than or equal to another PGF d' iff every coefficient of $X_1^{s_1} \cdots X_k^{s_k}$ in d is less than or equal to the corresponding coefficient in d'.

- $f \sqsubseteq_{L_k} g \iff \forall d \in D_k \colon f(d) \sqsubseteq_{D_k} g(d)$
  A PGF transformer is less than or equal to another iff it is pointwise less than or equal for all elements of D_k.

In the remaining thesis, we will leave out the index k of D_k and L_k if k is clear from context.

Lemma 3.1. *D_k and L_k are ω-complete partial orders.*

Proof. See Appendix **??**. □

Before we continue, we take a closer look at the sets D and L. Consider an element of D. Elements in D are restricted to having their absolute value and all coefficients between zero and one. Each of its coefficients is the probability that the corresponding program state appears. Neither of these probabilities can be more than one or less than zero, so these restrictions do not exclude any programs or PGFs we could use for describing actual programs.
Elements in L are functions from PGF to PGF. Every program transforms an initial PGF into its result. This means that all programs must correspond to an element in L: $[\![P]\!] \in L$.

3.3 Programs with Loops

Having introduced the two ω-complete partial orders D and L, we can go on defining the loop semantics $[\![P_W]\!]$ of a loop $P_W = \mathit{while}(B)\{P\}$. To do so, we define two functions. The first one is $\mathrm{wh}_{B,P}$ and it is defined as follows:

$$\mathrm{wh}_{B,P} \colon D \to D$$
$$\mathrm{wh}_{B,P}(G) = [\![P]\!]\left(\langle G\rangle_B\right)$$

$\mathrm{wh}_{B,P}$ applies the loop body once after restricting its argument to the loop condition. This corresponds to the decision if the loop body should be executed once more or not. $\mathrm{wh}_{B,P}$ chooses the parts that satisfy the condition and applies another iteration to them. We will later see that the remaining part ($\langle G\rangle_{\neg B}$) will form the semantics of the loop. We use $\mathrm{wh}_{B,P}$ very often in the remaining thesis. It always occurs in combination with a loop. In this case, if not stated otherwise, the index B will correspond to the loop condition and P to the loop body.
The second function is $\mathrm{H}_{B,P}$:

$$\mathrm{H}_{B,P} \colon L \to L,$$
$$\mathrm{H}_{B,P}(f)(G) = \langle G\rangle_{\neg B} + f\left(\mathrm{wh}_{B,P}(G)\right)$$

As for $\mathrm{wh}_{B,P}$, the indices B and P of $\mathrm{H}_{B,P}$ also correspond to the loop condition and loop body, respectively. Remember that L contains the set of all semantics functions of all programs. $\mathrm{H}_{B,P}$ is a map on L, so it basically transforms programs. In fact, $\mathrm{H}_{B,P}$ acts like a single iteration of the loop P_W to its argument. The loop body P is executed once unless the condition B does not hold. In this case, nothing is done. We can see that if we apply $\mathrm{H}_{B,P}$ to a program P':

$$\begin{aligned}
&\mathrm{H}_{B,P}([\![P']\!])(G) \\
&\qquad \text{(definition of } \mathrm{H}_{B,P}\text{)} \\
&= \langle G\rangle_{\neg B} + [\![P']\!]\left(\mathrm{wh}_{B,P}(G)\right) \\
&\qquad \text{(definition of } \mathrm{wh}_{B,P}\text{)} \\
&= \langle G\rangle_{\neg B} + [\![P']\!]\left([\![P]\!]\left(\langle G\rangle_B\right)\right) \\
&\qquad \text{(concatenation of programs)} \\
&= \langle G\rangle_{\neg B} + [\![P; P']\!]\left(\langle G\rangle_B\right) \\
&\qquad ([\![\mathit{skip}]\!] \text{ is the identity function}) \\
&= [\![\mathit{skip}]\!]\left(\langle G\rangle_{\neg B}\right) + [\![P; P']\!]\left(\langle G\rangle_B\right) \\
&\qquad (\mathit{if} \text{ semantics}) \\
&= [\![\mathit{if}(B)\,\{\,P; P'\,\}\,\mathit{else}\,\{\,\mathit{skip}\,\}]\!]\,(G)
\end{aligned}$$

Iterating $\mathrm{H}_{B,P}$ will mimic the behaviour of the loop P_W by unrolling the loop body. For example, applying $\mathrm{H}_{B,P}$ twice gives

$$\mathrm{H}^2_{B,P}([\![P']\!]) = [\![\mathit{if}(B)\,\{\,P; \mathit{if}(B)\,\{\,P; P'\,\}\,\mathit{else}\,\{\,\mathit{skip}\,\}\,\}\,\mathit{else}\,\{\,\mathit{skip}\,\}]\!]$$

which is the second unrolling of P_W. If we unroll $\mathrm{H}_{B,P}$ ad infinitum, we get a program equivalent to the loop P_W. In this case, unrolling ad infinitum is equivalent to taking the least fixed point of $\mathrm{H}_{B,P}$. We can find the least fixed point of $\mathrm{H}_{B,P}$ using Kleene's fixed point theorem.

Theorem 3.3 (Kleene's Fixed Point Theorem [**?**])**.** *Every Scott-continuous function F over an ω-complete partial order has a least fixed point which is $\sup_{n\in\mathbb{N}}\{F^n(\bot)\}$. $\bot$ is the bottom element of the partial order.*

To apply the theorem, we need to show that $\mathrm{H}_{B,P}$ is Scott-continuous.

Lemma 3.2. *Let B be a Boolean condition and P be a program. Then*

$$\mathrm{H}_{B,P}(f)(G) = \langle G\rangle_{\neg B} + f\left(\mathrm{wh}_{B,P}(G)\right) \text{ is Scott-continuous.}$$

Proof. See Appendix **??** □

$\mathrm{H}_{B,P}$ satisfies the preconditions of the theorem and we get the least fixed point F

$$F = \sup_{n\in\mathbb{N}} \left\{\mathrm{H}^n_{B,P}(\bot)\right\}$$

$\bot$ is the bottom element of L. It is the least element according to $\sqsubseteq_L$. $\sqsubseteq_L$ is defined pointwise using $\sqsubseteq_D$, so $\bot$ must always yield the least PGF, which is 0. Hence,

$$\bot(G) = \sum_{s\in\mathbb{Z}^k} 0 X_1^{s_1}\cdots X_k^{s_k}$$

$\mathrm{H}_{B,P}$ applies iterations of the loop P_W to its argument, so F is the semantics of the whole loop.

Definition 3.3 (Loop Semantics)**.** *Let B be a Boolean condition and P a program. The semantics of the loop $\mathit{while}\,(B)\{P\}$ is given by*

$$[\![\mathit{while}\,(B)\{P\}]\!] = F = \sup_{n\in\mathbb{N}} \left\{\mathrm{H}^n_{B,P}(\bot)\right\}$$

We now have a first characterisation of the semantics of P_W. $[\![P_W]\!]$ is the fixed point of unrolling the loop. Finding suprema by hand or software can be difficult, so we will go on to find more characterisations of $[\![P_W]\!]$. A closed form of $\mathrm{H}^n_{B,P}(\bot)$ is a first step.

Lemma 3.3. *Let B be a Boolean condition and P a program. Then*

$$\mathrm{H}^n_{B,P}(\bot)(G) = \sum_{i=0}^{n-1} \left\langle \mathrm{wh}^i_{B,P}(G)\right\rangle_{\neg B}$$

where $\mathrm{wh}_{B,P}(G) = [\![P]\!]\left(\langle G\rangle_B\right)$ *as defined above.*

Proof. See Appendix **??**. □

We now have another form for $\mathrm{H}_{B,P}$ and can express the loop semantics in another way.

$$[\![\mathit{while}\,(B)\{P\}]\!]\,(G) = \sup_{n\in\mathbb{N}} \left\{ \lambda G. \sum_{i=0}^{n-1} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \right\} (G)$$

Let us have a look at the sum inside the supremum. Each of its summands is a PGF with non-negative coefficients. Since PGFs are added coefficientwise, the coefficient of the whole sum for any $s \in \mathbb{Z}^k$ must keep increasing with n going to infinity. The supremum over all n of these sums must therefore be the infinite series.

Theorem 3.4. *Let B be a Boolean condition and P a program. Then*

$$[\![\mathit{while}\,(B)\{P\}]\!]\,(G) = \sum_{i=0}^{\infty} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B}$$

Proof. See Appendix **??**. □

The loop semantics is expressible as an infinite series of PGFs. Series are well understood and there is a fair amount of tools to deal with them.

Listing 3: Canonical example

```
X := 0;
F := 0;
while ( F = 0 ) {
  { X := X + 1 }[0.5]{ F := 1 }
}
```

Listing 4: Inverse canonical example

```
X := 0;
F := 0;
while ( F = 0 ) {
  { X := X - 1 }[0.5]{ F := 1 }
}
```

Figure 2: Two simple probabilistic programs.

4 Case Studies

Having introduced all elements of the semantics of our probabilistic language, we want to discuss several example programs. We will explain our semantics by applying it to each of the programs. All of the programs contain loops which are the most intricate language constructs.

4.1 Canonical Examples

First, we will discuss the examples in Figure **??**. They are very similar. Both contain a loop that runs as long as the variable F has the value 0. In the loop body, either the variable X is incremented/ decremented or F is set to 1 so that the loop terminates. Recall the semantics of a loop $\mathit{while}(B)\{P\}$:

$$[\![\mathit{while}(B)\{P\}]\!](G) = \sum_{i=0}^{\infty} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B}$$

In the first program, we have

- $B = (F = 0)$
- $P = \{X := X + 1\}\,[0.5]\,\{F := 1\}$.
- $G = 1$
- $\mathrm{wh}_{B,P}(D) = [\![P]\!](\langle D \rangle_B) = \frac{1}{2}[\![X := X + 1]\!](\langle D \rangle_B) + \frac{1}{2}[\![F := 1]\!](\langle D \rangle_B)$.

The first step to finding the loop semantics is finding a closed form for $\left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B}$. After applying the loop body manually two or three times, one can generalise that

$$\left\langle \mathrm{wh}^0_{B,P}(G) \right\rangle_{\neg B} = \langle 1 \rangle_{\neg B} = 0$$

$$\left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} = \left(\frac{1}{2}\right)^i X^{i-1} F^1, \text{ where } i > 0$$

We can verify this by an induction over i, see Appendix **??**. To get the complete loop semantics, we have to sum up these terms for $i \in \mathbb{N}$.

$$
\begin{aligned}
&[\![\mathit{while}\,(B)\{P\}]\!]\,(G) \\
=&\left\langle \mathrm{wh}^0_{B,P}\,(1)\right\rangle_{\neg B} + \sum_{i=1}^{\infty} \mathrm{wh}^i_{B,P}\,(1) \\
=&0 + \sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^i X^{i-1}F^1 \\
&\left(\text{factor out } \frac{1}{2}\right) \\
=&\frac{1}{2}\cdot\sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^{i-1} X^{i-1}F^1 \\
&(\text{transform indices}) \\
=&\frac{1}{2}\cdot\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^{i} X^{i}F^1
\end{aligned}
$$

The semantics resulting from the loop is an infinite sum, where X has value i with probability $\left(\frac{1}{2}\right)^{i+1}$. In fact, this is a geometric series and it has a closed form.

$$
[\![\mathit{while}\,(B)\{P\}]\!]\,(G) = \frac{1}{2}\cdot\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^{i} X^{i}F^1 = \frac{1}{2}\cdot\frac{F}{1-\frac{1}{2}X} = \frac{F}{2-X}
$$

When we apply the absolute value, we can see that the loop terminates with probability 1.

$$
\left|[\![\mathit{while}\,(B)\{P\}]\!]\,(G)\right| = \left|\frac{F}{2-X}\right| = 1
$$

Having found the semantics for the first program, we can go on with the next one. Calculating the semantics for the second program yields a similar result. Instead of being incremented, the variable X is decremented in the loop body. It will have negative values after more than one iteration of the loop. Again, we first find a closed form for $\left\langle \mathrm{wh}^i_{B,P}(1)\right\rangle_{\neg B}$. In this case, it holds:

$$
\begin{aligned}
\left\langle \mathrm{wh}^0_{B,P}\,(G)\right\rangle_{\neg B} &= \langle 1\rangle_{\neg B} = 0 \\
\left\langle \mathrm{wh}^i_{B,P}\,(G)\right\rangle_{\neg B} &= \left(\frac{1}{2}\right)^i X^{-i+1}F^1, \text{ where } i > 0
\end{aligned}
$$

The inductive proof is analogous to the one in Appendix **??** for the previous example. Note that we have a negative exponent for X. To get the loop semantics, we again have to sum up

Listing 5:

```
X := 0;
F := 0;
while ( F = 0 ) {
  {
    { X := X + 1 }[0.5]{ abort }
  }[0.5]{
    F := 1
  }
}
```

Figure 3: Example program with termination probability < 1.

all $\left\langle \mathrm{wh}^i_{B,P}(1) \right\rangle_{\neg B}$.

$$
\begin{aligned}
&[\![\mathit{while}\,(B)\{P\}]\!]\,(G) \\
=&\left\langle \mathrm{wh}^0_{B,P}(G) \right\rangle_{\neg B} + \sum_{i=1}^{\infty} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \\
=&0 + \sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^i X^{-i+1} F^1 \\
&\qquad \text{(transform indices)} \\
=&\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^{i+1} X^{-i} F^1 \\
&\qquad \left(\text{factor out } \frac{1}{2}\right) \\
=&\frac{1}{2} \cdot \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i X^{-i} F^1
\end{aligned}
$$

Again, the loop semantics is a geometric series with a closed form.

$$[\![\mathit{while}\,(B)\{P\}]\!]\,(G) = \frac{1}{2} \cdot \frac{F}{1 - \frac{1}{2}X^{-1}} = \frac{F}{2 - X^{-1}}$$

4.2 Termination Probability

Now, we will discuss the second example, the program in Figure **??**. The main difference to the other example programs is that it does not terminate with probability 1. We will derive a semantics and show that the loop halts with probability $\frac{2}{3}$. The loop does not halt with probability 1 because it contains an *abort* statement, so there are program runs that do not terminate at all. This affects the behaviour of the whole loop. For the program's loop $\mathit{while}\,(B)\{P\}$, we have the following setting:

- $B = (F = 0)$
- $P = \left\{\{X := X + 1\}[\frac{1}{2}]\{abort\}\right\}[\frac{1}{2}]\{F := 1\}$
- $G = 1$
- $\begin{aligned}
&\mathrm{wh}_{B,P}(D) = [\![P]\!]\left(\langle D\rangle_B\right)\\
=&\frac{1}{2}\left(\frac{1}{2}[\![X := X + 1]\!](D) + \frac{1}{2}[\![abort]\!](D)\right) + \frac{1}{2}[\![F := 1]\!](D)\\
=&\frac{1}{2}\left(\frac{1}{2}[\![X := X + 1]\!](D) + 0\right) + \frac{1}{2}[\![F := 1]\!](D)\\
=&\frac{1}{4}[\![X := X + 1]\!](D) + \frac{1}{2}[\![F := 1]\!](D)
\end{aligned}$

The way to finding the loop semantics leads over finding a closed form for $\left\langle \mathrm{wh}^i_{B,P}(G)\right\rangle_{\neg B}$. Doing this gives:

$$\left\langle \mathrm{wh}^0_{B,P}(G)\right\rangle_{\neg B} = \langle 1\rangle_{\neg B} = 0$$

$$\left\langle \mathrm{wh}^i_{B,P}(G)\right\rangle_{\neg B} = \frac{1}{2}\cdot\left(\frac{1}{4}\right)^{i-1} X^{i-1}F^1, \text{ where } i > 0$$

A proof can be found in Appendix **??**. The term is quite similar to the very first example, where we had

$$\left\langle \mathrm{wh}^i_{B,P}(G)\right\rangle_{\neg B} = \left(\frac{1}{2}\right)^{i} X^{i-1}F^1$$

The difference is that the exponentiated constant is not $\frac{1}{2}$ but $\frac{1}{4}$ and there is a constant factor of $\frac{1}{2}$ in front. The loop semantics is again a geometric series although there are the nested

probabilistic choices and an *abort* statement.

$$
\begin{aligned}
&[\![\mathit{while}\,(B)\{P\}]\!]\,(G)\\
=&\left\langle \mathrm{wh}^0_{B,P}(G)\right\rangle_{\neg B} + \sum_{i=1}^{\infty}\left\langle \mathrm{wh}^i_{B,P}\,(1G)\right\rangle_{\neg B}\\
=&0+\sum_{i=1}^{\infty}\frac{1}{2}\cdot\left(\frac{1}{4}\right)^{i-1}X^{i-1}F^1\\
&\qquad\text{(factor out, transform indices)}\\
=&\frac{1}{2}\cdot\sum_{i=0}^{\infty}\left(\frac{1}{4}\right)^{i}X^{i}F^1\\
&\qquad\text{(closed form of geometric series)}\\
=&\frac{1}{2}\cdot\frac{F}{1-\frac{1}{4}X}\\
=&\frac{F}{2-\frac{1}{2}X}
\end{aligned}
$$

Applying the absolute value to the loop semantics gives us the termination probability.

$$
\left|[\![\mathit{while}\,(B)\{P\}]\!]\,(G)\right| = \left|\frac{F}{2-\frac{1}{2}X}\right| = \frac{1}{2-\frac{1}{2}} = \frac{2}{3}
$$

We see that the loop does not terminate with probability 1.

4.3 Two Equivalent Programs

The two programs in Figure **??** are parametrised with p and q. Both variables only occur in the probabilistic choices of the programs, so they are restricted to values in $[0,1]$. In order to execute a program, a value for both variables must be chosen. However, our semantics can be applied leaving p and q in place which gives a more general result about the program's behaviour. The programs were introduced by Kiefer et al. [**?**]. At the time, they were proven to be equivalent for $p=\frac{1}{2}$ and $q=\frac{2}{3}$. Later, Gretz et al. [**?**] proved that the expected value of the variable X is equivalent for all combinations of p and q satisfying $q=\frac{1}{2-p}$. We will extend this result and show that not only the expected value of X is the same, but even the underlying distributions are equivalent for these combinations of p and q.

Consecutive Loops

The first program is basically the concatenation of the two canonical examples. It contains two loops, we will call them L^+ and L^- because they respectively increment or decrement the

Listing 6: Consecutive loops.

```
X := 0;
F := 0;
while ( F = 0 ) {
  { X := X + 1 }[p]{ F := 1 }
};

F := 0;
while ( F = 0 ) {
  { X := X - 1 }[q]{ F := 1 }
}
```

Listing 7: Separated loops.

```
X := 0;
{ F := 0 }[0.5]{ F := 1 };
if ( F = 0 ) {
  while ( F = 0 ) {
    { X := X + 1 }[p]{ F := 1 }
  }

} else {
  F := 0;
  while ( F = 0 ) {
    X := X - 1;
    { skip }[q]{ F := 1 }
  }
}
```

Figure 4: Example programs from Kiefer et al. [**?**].

variable X. L^+ generates the following geometric distribution over the positive integers:

$$[\![L^+]\!]\,(1) = (1-p)\cdot\sum_{i=0}^{\infty} p^i X^i F^1$$

A proof can be found in Appendix **??**. The factor $1-p$ comes from taking the right branch of the probabilistic choice which sets F to 1. This only happens once while the left branch can be taken arbitrarily often. That is why p^i appears as a factor in the sum. If X has the value i, the left branch was taken i times, leading to the exponentiation of p. The next statement resets F to 0, so now it remains to calculate

$$[\![L^-]\!]\left([\![F := 0; L^+]\!]\,(1)\right) = [\![L^-]\!]\left((1-p)\cdot\sum_{i=0}^{\infty} p^i X^i F^0\right)$$

To simplify the calculation and the upcoming comparison of the two results, we will now switch to writing the semantics using tuples. We have two variables, so the tuples have four entries. We can reduce this by having a look at the variables. Contrary to X, F is only ever assigned 0 or 1, so F is never negative. X can have positive and negative valuations, because it is incremented and decremented. The entries of the semantics tuples where F is negative are thus always 0, so we can as well leave them out. This reduces the tuples to two entries. We assign the non-negative valuations of X to the first entry and the negative ones to the second entry. Unspectacularly, the semantics from above becomes

$$[\![L^-]\!]\left((1-p)\cdot\sum_{i=0}^{\infty} p^i X^i F^0, 0\right)$$

The loop L^- decrements X in every iteration except its last, so after every iteration a new summand will be added to the second entry of the tuple. If the loop terminates after one

iteration, no decrement took place and F was immediately set to 1. If the loop terminates after two iterations, X was decremented once to -1 and afterwards F was set to 1. After n iterations, the minimal value of X is $-(n-1)$. This behaviour is expressed in $\mathrm{wh}^n_{B,P}$. Its general form is

$$\begin{aligned}&\left\langle \mathrm{wh}^n_{B,P}\left((1-p)\cdot\sum_{i=0}^{\infty} p^i X^i F^0, 0\right)\right\rangle_{\neg B}\\ =&(1-q)\cdot q^{n-1}\cdot(1-p)\cdot\left(\sum_{i=0}^{\infty} p^{i+(n-1)}X^iF^1, \sum_{i=1}^{n-1} p^{(n-1)-i}X^{-i}F^1\right)\end{aligned}$$

The factor $1-p$ is still there from the loop L^+. As above, $1-q$ stems from taking the right branch of the probabilistic choice once and q^{n-1} from taking the left branch the other times. The minimal negative valuation of X is $-(n-1)$. In the first tuple, the factors do not start with p^0 for i equal to zero, but with p^{i+n-1}. This happens because lower values of X are more likely to be decremented into the negative range than higher ones. Building the infinite series of all these terms gives the semantics of the whole program:

$$\begin{aligned}&[\![L^+; F := 0; L^-]\!]\,(1,0)\\ &\qquad\text{(semantics of concatenation)}\\ =\,&[\![L^-]\!]\,\big([\![F := 0; L^-]\!]\,(1,0)\big)\\ &\qquad\text{(semantics of the first loop } L^+\text{)}\\ =\,&[\![L^-]\!]\left((1-p)\cdot\sum_{i=0}^{\infty} p^i X^i F^0, 0\right)\\ &\qquad\text{(semantics of a loop)}\\ =\,&\sum_{n=0}^{\infty}\left\langle \mathrm{wh}^n_{B,P}\left((1-p)\cdot\sum_{i=0}^{\infty} p^i X^i F^0, 0\right)\right\rangle_{\neg B}\\ &\qquad\text{(simplify series, proof in Appendix ??)}\\ =\,&\frac{(1-q)\cdot(1-p)}{1-qp}\cdot\left(\sum_{i=0}^{\infty} p^iX^iF^1, q\cdot\sum_{i=0}^{\infty} q^iX^{-(i+1)}F^1\right)\end{aligned}$$

Separated Loops

Finding the second program's semantics is easier, because the two loops are in different branches of a conditional choice, so the second one does not operate on the result of the first one, but on a simpler PGF. As above, there is one loop that increments and one that decrements X. We will also name them L^+ and L^-, respectively. L^- is slightly different than in the first program. Here, X is decremented in every iteration of the loop and only afterwards the decision to either loop again or terminate the loop is made. Before treating the loops, we

have to take care of the conditional choice and the probabilistic choice preceding it. In it, F is set to 0 or 1 with probability $\frac{1}{2}$ each. Then L^+ is executed if F is 0 and L^- is executed otherwise. In fact, this is equivalent to the probabilistic choice

$$\left\{L^+\right\}[0.5]\left\{L^-\right\}$$

For the semantics of the whole program, we will have to calculate

$$\frac{1}{2}\cdot\left[\!\left[L^+\right]\!\right](1,0)+\frac{1}{2}\cdot\left[\!\left[L^-\right]\!\right](1,0)$$

The first loop L^+ is the same as in the first canonical program, so we already know its semantics.

$$\frac{1}{2}\cdot\left[\!\left[L^+\right]\!\right](1,0)=\frac{1}{2}\cdot\left((1-p)\cdot\sum_{i=0}^{\infty}p^iX^iF^1,0\right)$$

For the loop L^-, we find that after n iterations, the minimal value of X is $-n$. We can see this in $\left\langle \mathrm{wh}^i_{B,P}(1,0)\right\rangle_{\neg B}$ (proof in Appendix **??**):

$$\left\langle \mathrm{wh}^i_{B,P}(1,0)\right\rangle_{\neg B}=(1-q)\cdot\left(0,q^{i-1}X^{-i}F^1\right)$$

The sum of all these terms gives us the loop semantics.

$$\begin{aligned}
&\frac{1}{2}\cdot\left[\!\left[L^-\right]\!\right](1,0)\\
=&\frac{1}{2}\cdot\sum_{i=0}^{\infty}\left\langle \mathrm{wh}^i_{B,P}(1,0)\right\rangle_{\neg B}\\
=&\frac{1}{2}\cdot\sum_{i=1}^{\infty}(1-q)\cdot\left(0,q^{i-1}X^{-i}F^1\right)\\
&\qquad\text{(factor out, transform indices)}\\
=&\frac{1}{2}\cdot\left(0,(1-q)\cdot\sum_{i=0}^{\infty}q^iX^{-(i+1)}F^1\right)
\end{aligned}$$

We now know the semantics of both loops, so we add them to get the semantics of the whole program.

$$
\begin{aligned}
&\frac{1}{2} \cdot [\![L^+]\!]\,(1,0) + \frac{1}{2} \cdot [\![L^-]\!]\,(1,0) \\
&\qquad \text{(substitute results)} \\
=&\frac{1}{2} \cdot \left((1-p) \cdot \sum_{i=0}^{\infty} p^i X^i F^1, 0 \right) + \frac{1}{2} \cdot \left(0, (1-q) \cdot \sum_{i=0}^{\infty} q^i X^{-(i+1)} F^1 \right) \\
=&\frac{1}{2} \cdot \left((1-p) \cdot \sum_{i=0}^{\infty} p^i X^i F^1, (1-q) \cdot \sum_{i=0}^{\infty} q^i X^{-(i+1)} F^1 \right) \\
&\qquad \left(\text{distribute } \frac{1}{2}\right) \\
=&\left(\frac{1-p}{2} \cdot \sum_{i=0}^{\infty} p^i X^i F^1, \frac{1-q}{2} \cdot \sum_{i=0}^{\infty} q^i X^{-(i+1)} F^1 \right)
\end{aligned}
$$

Equivalence

After determining the semantics for both programs, we will now find out the values of p and q for which they are equivalent. Remember semantics (1) and (2) for their programs:

$$\frac{(1-q)\cdot(1-p)}{1-qp} \cdot \left(\sum_{i=0}^{\infty} p^i X^i F^1, q \cdot \sum_{i=0}^{\infty} q^i X^{-(i+1)} F^1 \right) \tag{1}$$

$$\left(\frac{1-p}{2} \cdot \sum_{i=0}^{\infty} p^i X^i F^1, \frac{1-q}{2} \cdot \sum_{i=0}^{\infty} q^i X^{-(i+1)} F^1 \right) \tag{2}$$

One can see that sums in the corresponding entries of both tuples are the same. They only differ in the constants they are multiplied by. In consequence, we can simply equate them and solve for q. Because the tuples have two entries, we get two equations.

$$\frac{(1-q)\cdot(1-p)}{1-qp} = \frac{1-p}{2} \tag{1}$$

$$\frac{(1-q)\cdot(1-p)\cdot q}{1-qp} = \frac{1-q}{2} \tag{2}$$

Solving these for q gives the same result in both cases. Hence, extending the results of Gretz et al. in [**?**], we proved that the two programs are equal in distribution for

$$q = \frac{1}{2-p}, \text{ where } p, q \neq 1 \ .$$

5 Bisimulations

Introduction

In [**?**], Arbab et al. introduced the concept of *coinduction*. Coinduction can be used to reason about sequences of numbers using relations called *bisimulations*. Inspired by their ideas, we will introduce two methods that use relations of PGFs to reason about loop semantics.

Definition 5.1 (Weak Bisimulation)**.** *Let B be a Boolean condition and P a program. A* weak bisimulation *for B and P is a binary relation $R \subseteq D \times D$ such that for all G, $K \in D$, if $(K, G) \in R$, then*

Let $G' = \mathrm{wh}_{B,P}(G)$

i) $\langle G' \rangle_{\neg B} \sqsubseteq_D K$

ii) $(K - \langle G' \rangle_{\neg B}, G') \in R$

The definition for a weak bisimulation is an inductive one. Assume the pair (K, G) of PGFs is an element of a bisimulation R. Using the property **??**, we can find another pair of PGFs that is also an element of R. Note that the first and second pair are not necessarily distinct. The entries K and G correspond to output and input of a loop. K is a candidate for the semantics of a loop, G is the loop's input. Recall the loop semantics:

$$[\![\mathit{while}\,(B)\{P\}]\!]\,(G) = \sum_{i=0}^{\infty} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B}$$

To get a statement about the loop semantics, we can repeatedly apply **??** to get new pairs of PGFs. In ii), the second entry of the new pair is $\mathrm{wh}_{B,P}(G)$. As we have seen before, $\mathrm{wh}_{B,P}$ applies the loop body P once to its input. Hence, every application of ii) applies the loop body P once to G. This means that applying ii) is similar to unrolling the loop once. In the first entry, $\langle G' \rangle_{\neg B} = \langle \mathrm{wh}_{B,P}(G) \rangle_{\neg B}$ is subtracted from K in an application of ii). The subtractions cumulate with every application of ii) so that iterating ii) ad infinitum subtracts the semantics of the loop from K. Unfortunately, a weak bisimulation can only prove that K is an overapproximation of the loop semantics.

Theorem 5.1. *Let B be a boolean condition, P a program and R a basic bisimulation. Then*

$$(K, G) \in R \wedge G \models B \implies [\![\mathit{while}\,(B)\{P\}]\!]\,(G) \sqsubseteq_D K$$

Proof. See Appendix **??**. □

Here, G satisfies B if the restriction $\langle G \rangle_B$ is the same as G:

$$G \models B \iff \langle G \rangle_B = G \iff \langle G \rangle_{\neg B} = 0$$

In other words, every program state for which the corresponding coefficient is non-zero must satisfy the condition B. This part of the precondition restricts the loops we can handle to those where the input PGF satisfies the loop condition. In practice, many loops appearing have that property. For example, in all loops in the previous examples, their input PGF

Listing 8: Canonical example

```
X := 0
F := 0
while (F = 0) {
  {X := X + 1}[0.5]{F := 1}
}
```

Figure 5: A probabilistic program.

satisfies the loop condition.
In a weak bisimulation, K is an overapproximation for the loop semantics. The reason is that $\langle \mathrm{wh}_{B,P}(G) \rangle_{\neg B}$ in property **??** may be 0. For example, $R_0 = \{(1,0)\}$ is a valid weak bisimulation. Applying ii) gives again $(1,0)$. There is no loop P_W such that $[\![P_W]\!](0) = 1$ yet for all loops it holds that $[\![\mathit{while}\,(B)\{P\}]\!](0) \sqsubseteq_D 1$. In order to prove that a pair (K,G) in a bisimulation is in fact the output and input of a loop, we need to strengthen the definition of a weak bisimulation by another property.

Definition 5.2 (Strong Bisimulation)**.** *Let B be a boolean condition and P a program. An* advanced bisimulation *for B and P is a binary relation $R \subseteq D \times D$ that is a basic bisimulation with one additional property. For all $G, K \in D$ and a constant $\varepsilon \in (0,1)$, if $(K,G) \in R$, then*

Let $G' = \mathrm{wh}_{B,P}(G)$

i) $\langle G' \rangle_{\neg B} \sqsubseteq_D K$

ii) $(K - \langle G' \rangle_{\neg B}, G') \in R$

iii) $|\langle G' \rangle_{\neg B}| \geq \varepsilon \cdot |K|$

The strong bisimulation introduces a constant ε which is the same all over R. This means that we cannot choose ε after every application of **??**. ε must be in the interval $(0,1)$ so that the first and third properties can never contradict: **??** implies $|\langle G' \rangle_{\neg B}| \leq |K|$. With ε chosen greater than 1, this would be a contradiction to **??**.
Of course, a strong bisimulation is also a weak bisimulation because it satisfies properties **??** and **??**. The third property iii) prevents the problem we had with weak bisimulations. It guarantees that K is strictly decreased in every application of ii). In consequence, a pair (K,G) in a strong bisimulation is a pair of output and input of a loop.

Theorem 5.2. *Let B be a boolean condition, P a program and R a strong bisimulation for B and P with constant ε. Then*

$$(K,G) \in R \wedge G \models B \implies K = [\![\mathit{while}\,(B)\{P\}]\!](G)$$

Proof. See Appendix **??**. □

Case study

As an example, we will find a bisimulation for the program in Figure **??**. Define $B = (F = 0)$ as the loop condition and $P = \{X := X + 1\}[0.5]\{F := 1\}$ as the loop body. The input of the

loop is $G = 1$. The loop's output is $K := \frac{1}{2} \cdot \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i X^i F^1$, as we have seen in Section **??**. We now want to apply the previous theorem to the loop. To do so, we define a relation R such that (K, G) is an element of R and prove that R is a strong bisimulation.

$$R = \{(K, G)\} \cup \left\{ \left(\frac{1}{2} \cdot \sum_{i=n}^{\infty} \left(\frac{1}{2}\right)^i X^i F^1, \left(\frac{1}{2}\right)^n X^n F^0 + \left(\frac{1}{2}\right)^n X^{n-1} F^1 \right) \middle| n \in \mathbb{N}_{>0} \right\}$$

Apparently, (K, G) is an element of R, so the pair of output and input of the loop are in R. It remains to prove that R is a strong bisimulation.

Proof. We have to show the properties **??**, **??** and **??** for every element of R. We do this by showing them once for (K, G) and then for every $n > 0$. During the proof, we will also have to find a value for ε.

- Property i) for (K, G):

$$\begin{aligned} \langle \mathrm{wh}_{B,P}(G) \rangle_{\neg B} &= \left\langle \frac{1}{2} X^1 F^0 + \frac{1}{2} X^0 F^1 \right\rangle_{\neg B} = \frac{1}{2} X^0 F^1 \\ &\sqsubseteq_D \frac{1}{2} X^0 F^1 + \frac{1}{4} X^1 F^1 + \frac{1}{8} X^2 F^1 + \dots \\ &= K \end{aligned}$$

- Property ii) for (K, G):

$$\begin{aligned} &\left(K - \langle \mathrm{wh}_{B,P}(G) \rangle_{\neg B}, \mathrm{wh}_{B,P}(G)\right) \\ &= \left(K - \frac{1}{2} X^0 F^1, \frac{1}{2} X^1 F^0 + \frac{1}{2} X^0 F^1 \right) \\ &= \left(\frac{1}{2} \cdot \sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^i X^i F^1, \frac{1}{2} X^1 F^0 + \frac{1}{2} X^0 F^1 \right) \\ &\in R \text{ for } n = 1 \end{aligned}$$

- Property iii) for (K, G):

$$\begin{aligned} \left| \langle \mathrm{wh}_{B,P}(G) \rangle_{\neg B} \right| &= \left| \frac{1}{2} X^0 F^1 \right| = \frac{1}{2} \\ \text{and } |K| &= \left| \frac{1}{2} \cdot \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i X^i F^1 \right| = \frac{1}{2} \cdot \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = 1 \\ \implies \left| \langle \mathrm{wh}_{B,P}(G) \rangle_{\neg B} \right| &= \frac{1}{2} \geq \varepsilon \cdot 1 = |K| \text{ for } \varepsilon = \frac{1}{2} \end{aligned}$$

In the proof for property iii), we chose $\varepsilon = \frac{1}{2}$. We will have to use the same value for the remaining cases or find a new value that fits all cases.

- Property i) for any $n > 0$:

$$\begin{aligned}
&\left\langle \mathrm{wh}_{B,P}\left(\left(\frac{1}{2}\right)^n X^n F^0 + \left(\frac{1}{2}\right)^n X^{n-1} F^1\right)\right\rangle_{\neg B} \\
=&\left\langle \left(\frac{1}{2}\right)^{n+1} X^{n+1} F^0 + \left(\frac{1}{2}\right)^{n+1} X^n F^1 \right\rangle_{\neg B} \\
=&\left(\frac{1}{2}\right)^{n+1} X^n F^1 \\
\sqsubseteq_D &\left(\frac{1}{2}\right)^{n+1} X^n F^1 + \left(\frac{1}{2}\right)^{n+2} X^{n+1} F^1 + \ldots \\
=&\frac{1}{2} \cdot \sum_{i=n}^{\infty} \left(\frac{1}{2}\right)^i X^i F^1
\end{aligned}$$

- Property ii) for any $n > 0$: First of all, let

$$\begin{aligned}
G_{n+1} &:= \mathrm{wh}_{B,P}\left(\left(\frac{1}{2}\right)^n X^n F^0 + \left(\frac{1}{2}\right)^n X^{n-1} F^1\right) \\
&= \left(\frac{1}{2}\right)^{n+1} X^{n+1} F^0 + \left(\frac{1}{2}\right)^{n+1} X^n F^1
\end{aligned}$$

  G_{n+1} appears in both entries of the tuple in property ii):

$$\begin{aligned}
&\left(\frac{1}{2} \cdot \sum_{i=n}^{\infty} \left(\frac{1}{2}\right)^i X^i F^1 - \langle G_{n+1}\rangle_{\neg B}, G_{n+1}\right) \\
=&\left(\frac{1}{2} \cdot \sum_{i=n}^{\infty} \left(\frac{1}{2}\right)^i X^i F^1 - \left(\frac{1}{2}\right)^{n+1} X^n F^1, G_{n+1}\right) \\
=&\left(\frac{1}{2} \cdot \sum_{i=n+1}^{\infty} \left(\frac{1}{2}\right)^i X^i F^1, \left(\frac{1}{2}\right)^{n+1} X^{n+1} F^0 + \left(\frac{1}{2}\right)^{n+1} X^n F^1\right) \\
&\in R \text{ for } n+1
\end{aligned}$$

- Property iii) for any $n > 0$: For the first entries of elements in R, the following holds:

$$\begin{aligned}
&\left| \frac{1}{2} \cdot \sum_{i=n}^{\infty} \left(\frac{1}{2}\right)^i X^i F^1 \right| \\
=&\frac{1}{2} \cdot \sum_{i=n}^{\infty} \left(\frac{1}{2}\right)^i \\
=&\frac{1}{2} \cdot \left(\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i - \sum_{i=0}^{n-1} \left(\frac{1}{2}\right)^i \right) \\
=&\left(\frac{1}{2}\right)^n
\end{aligned}$$

  For the second entries, we derive the following:

$$\begin{aligned}
&\left| \left\langle \mathrm{wh}_{B,P} \left(\left(\frac{1}{2}\right)^n X^n F^0 + \left(\frac{1}{2}\right)^n X^{n-1} F^1 \right) \right\rangle_{\neg B} \right| \\
=&\left| \left(\frac{1}{2}\right)^{n+1} X^n F^1 \right| \\
=&\left(\frac{1}{2}\right)^{n+1}
\end{aligned}$$

  We see that for $\varepsilon = \frac{1}{2}$ the property is satisfied. Note that we do not have to change the value of ε that we chose earlier.

$$\begin{aligned}
&\left| \left\langle \mathrm{wh}_{B,P}(\mathrm{wh}^n_{B,P}(G)) \right\rangle_{\neg B} \right| \\
=&\left(\frac{1}{2}\right)^{n+1} \geq \varepsilon \left(\frac{1}{2}\right)^n \\
=&\left| \sum_{i=n}^{\infty} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \right|
\end{aligned}$$

□

We have shown that R is a strong bisimulation for B and P. Since $G \models B$, we can apply Theorem **??** and Theorem **??** to the loop from above. The first thing we get is:

$$[\![\mathit{while}\,(B)\{P\}]\!]\,(G) \sqsubseteq_D K$$

and secondly, the stronger version:

$$[\![\mathit{while}\,(B)\{P\}]\!]\,(G) = K$$

6 Semantics Equivalence

We want to prove that our semantics is equivalent to an already established one. We chose McIver and Morgan's *weakest preexpectation semantics* [**?**], for short wp-semantics. The underlying programming language they use is basically the same as in this work. In addition to the seven operations we use in our programming language, it contains a nondeterministic choice operator. Since we lack that operator, we will prove the equivalence of all programs that do not use it. While our semantics transforms PGFs, the wp-semantics transforms expectations.

Definition 6.1 (Expectation [**?**])**.** *An* expectation f *is a map from the state space* $\mathbb{S}$ *into the nonnegative reals. The set* E *of all expectations is*

$$E = \{\mathbb{S} \to \mathbb{R}_{\geq 0}\}$$

In our case, the set of all program states is $\mathbb{S} = \mathbb{Z}^k$. An expectation can be given as an expression over the program variables. The wp-semantics itself is defined by inductively on the operations in the programming language.

Definition 6.2 (wp-Semantics [**?**])**.** *Let* B *be a Boolean condition,* P *and* Q *programs and* f *an expectation. The weakest preexpectation semantics* wp *of a program is defined as follows:*

1. $\mathrm{wp}\,(\mathit{skip}, f) = f$.
2. $\mathrm{wp}\,(\mathit{abort}, f) = 0$.
3. $\mathrm{wp}\,(X := e, f) = f[X/e]$.
4. $\mathrm{wp}\,(P; Q, f) = \mathrm{wp}\,(P, \mathrm{wp}\,(Q, f))$.
5. $\mathrm{wp}\,(\mathit{if}(B)\,\{\,P\,\}\,\mathit{else}\,\{\,Q\,\}, f) = [B] \cdot \mathrm{wp}\,(P, f) + [\neg B] \cdot \mathrm{wp}\,(Q, f)$.
6. $\mathrm{wp}\,(\{P\}[p]\{Q\}, f) = p \cdot \mathrm{wp}\,(P, f) + (1-p) \cdot \mathrm{wp}\,(Q, f)$.
7. $\mathrm{wp}\,(\mathit{while}\,(B)\{P\}, f) = \mu X.\,([B] \cdot \mathrm{wp}\,(P, X) + [\neg B] \cdot f)$.

In **??**, 0 is the null expectation $f(s) = 0$ for any program state s. The expression $f[X/e]$ in **??** means that every occurrence of X in f is replaced with e. When f is given as an expression over the program variables, this replaces X with its assigned value. In **??** and **??**, $[B]$ is a function from the state space to $\{0, 1\}$:

$$[B](s) = \begin{cases} 0 & \text{if } s \models B \\ 1 & \text{otherwise} \end{cases}$$

$[B]$ is multiplied by an expectation f, so that $([B] \cdot f)(s) = [B](s) \cdot f(s)$. This nulls the expectation for all program states that do not satisfy the condition and keeps the expectation's value everywhere else. It resembles the restriction $\langle G \rangle_B$ that we introduced for our semantics. The wp-semantics of the loop is also defined by a least fixed point. μ is the fixed point operator. It returns the least fixed point of the function $\mathrm{ww}_f(X) = [B] \cdot \mathrm{wp}\,(P, X) + [\neg B] \cdot f$ iterated on the null expectation.
In order to prove the equivalence of the two semantics, we will show that the expected value of an expectation is the same in both semantics.

Definition 6.3. *Let* $\mu\colon \mathbb{S} \to [0,1]$ *be a measure and* f *an expectation. The* expected value $\mathbb{E}_\mu(f)$ *is defined as follows.*

$$\mathbb{E}_\mu(f) = \sum_{s \in \mathbb{S}} \mu(s) \cdot f(s)$$

A measure assigns a probability to every program state. Let f be an expectation. The expected value of f after executing a program P can be expressed in both semantics. Given a measure μ, executing P in the wp-semantics means transforming f and then calculating the expected value:

$$\mathbb{E}_\mu(\mathrm{wp}\,(P, f))$$

μ corresponds to the initial distribution of the variables and $\mathrm{wp}\,(P, f)$ corresponds to the execution of P. For example, in the programs from the previous chapters, it was always

$$\mu(s) = \begin{cases} 1 & \text{if } s = (0, \ldots, 0) \\ 0 & \text{otherwise} \end{cases}$$

Since our semantics transforms PGFs and not expectations, the expectation after the execution must be expressed differently. A PGF is a formal power series whose coefficients correspond to the probability of program state encoded in the exponents of the variables. Thus, for every PGF there is a corresponding measure.

Theorem 6.1. *For every PGF* $G = \sum_{s \in \mathbb{Z}^k} \mu_s X_1^{s_1} \cdots X_k^{s_k}$ *there is a corresponding measure* μ_G*.* $\mu_G(s_1, \ldots, s_k)$ *equals the coefficient of* $X_1^{s_1} \cdots X_k^{s_k}$*.*

Transforming a PGF means transforming a measure. We can now express what we are looking for:

$$\mathbb{E}_{[\![P]\!](\mu)}(f)$$

Here, $[\![P]\!](\mu)$ means that μ is transformed according to the same rules that we would transform its corresponding PGF with. We will go on to prove that both versions of the expectation after execution have the same value.

Theorem 6.2. *Given a program* P*, a measure* μ *and an expectation* f*, the following holds:*

$$\mathbb{E}_{[\![P]\!](\mu)}(f) = \mathbb{E}_\mu(\mathrm{wp}\,(P, f))$$

Proof. Proof by induction over the program structure.
Base Cases:

(*skip*)

$$\begin{aligned}
&\mathbb{E}_{[\![skip]\!](\mu)}(f) \\
&\qquad \text{(definition expected value)} \\
=&\sum_{s\in\mathbb{S}} [\![skip]\!](\mu)(s)\cdot f(s) \\
&\qquad \text{(PGF } \textit{skip} \text{ semantics)} \\
=&\sum_{s\in\mathbb{S}} \mu(s)\cdot f(s) \\
=&\mathbb{E}_{\mu}(f) \\
&\qquad \text{(wp } \textit{skip} \text{ semantics)} \\
=&\mathbb{E}_{\mu}(\mathrm{wp}\,(skip, f))
\end{aligned}$$

(*abort*)

$$\begin{aligned}
&\mathbb{E}_{[\![abort]\!](\mu)}(f) \\
=&\sum_{s\in\mathbb{S}} [\![abort]\!](\mu)(s)\cdot f(s) \\
&\qquad \text{(PGF } \textit{abort} \text{ semantics)} \\
=&\sum_{s\in\mathbb{S}} 0\cdot f(s) \\
=&\sum_{s\in\mathbb{S}} \mu(s)\cdot 0 \\
&\qquad \text{(wp } \textit{abort} \text{ semantics)} \\
=&\sum_{s\in\mathbb{S}} \mu(s)\cdot \mathrm{wp}\,(abort, f)\,(s) \\
=&\mathbb{E}_{\mu}(\mathrm{wp}\,(abort, f))
\end{aligned}$$

($X_j := e$) Given an index $1 \leq j \leq k$ of a variable, a program state $s \in \mathbb{S}$ and an expression e, let

$$I_j^s(e) = \{(s_1, \ldots, s_{j-1}, v, s_{j+1}, \ldots, s_k) \mid e(s_1, \ldots, v, \ldots, s_k) = s_j\}$$

$I_j^s(e)$ is the set of all program states that are mapped to the state s by the semantics of the assignment $X_j := e$.
For example, given the PGF $G = \frac{1}{2}X_1^0X_2^1 + \frac{1}{3}X_1^0X_2^4$ and the assignment $X_2 := 2$. Then $[\![X_2 := 2]\!]$ maps both distinct states of G with their probabilities to one state with combined probability:

$$[\![X_2 := e]\!](G) = \frac{1}{2}X_1^0X_2^2 + \frac{1}{3}X_1^0X_2^2 = \frac{5}{6}X_1^0X_2^2 = \left(\sum_{s\in I_2^{0,2}(2)} \mu(s)\right)\cdot X_1^0X_2^2$$

Note that $I_j^s(e)$ can also be the empty set, for example with $j = 2$ and $s = (0, 4)$ in the example above.
One can see that I_j^s can be used to describe the terms that coincide after a variable assignment. We will use I_j^s to factor out and group terms in the next step.
Therefore, consider now the general assignment $X_j := e$ which is dealt with in this base case of the induction. As above, the assignment may map some distinct states to the same state, so we can also use $I_j^s(e)$ for appropriate j and s in the expected value.

$$\begin{aligned}
&\mathbb{E}_{[\![X_j := e]\!](\mu)}(f) \\
=&\sum_{s \in \mathbb{S}} [\![X_j := e]\!](\mu)(s) \cdot f(s) \\
&\qquad \left(\text{replace with combined probabilities using } I_j^s\right) \\
=&\sum_{s \in \mathbb{S}} \left(\sum_{s' \in I_j^s(e)} \mu(s') \right) \cdot f(s)
\end{aligned}$$

A similar approach works for the wp semantics. Starting with the semantics of the assignment, we get the following.

$$\mathrm{wp}\,(X_j := e, f)\,(s) = f[X_j/e](s) = f(s_1, \ldots, s_{j-1}, e(s), s_{j+1}, \ldots, s_k)$$

As before, there can be distinct program states s for which $f(s_1, \ldots, s_{j-1}, e(s), s_{j+1}, \ldots, s_k)$ is the same. In the sum of the expected value, these can be factored out and combined by using $I_j^s(e)$.

$$\begin{aligned}
&\mathbb{E}_{\mu}(\mathrm{wp}\,(X_j := e, f)) \\
=&\sum_{s \in \mathbb{S}} \mu(s) \cdot \mathrm{wp}\,(X_j := e, f)\,(s) \\
=&\sum_{s \in \mathbb{S}} \mu(s) \cdot f(s_1, \ldots, s_{j-1}, e(s), s_{j+1}, \ldots, s_k) \\
&\qquad (\text{factor out all } f(s) \text{ that coincide}) \\
=&\sum_{s \in \mathbb{S}} \left(\sum_{s' \in I_j^s(e)} \mu(s') \right) \cdot f(s)
\end{aligned}$$

Now it is apparent that both expected values are equal.

Induction Hypothesis:
Let P, Q be programs such that for all measures μ and expectations f the following holds:

$$\mathbb{E}_{[\![P]\!](\mu)}(f) = \mathbb{E}_{\mu}(\mathrm{wp}\,(P, f))$$

$$\mathbb{E}_{[\![Q]\!](\mu)}(f) = \mathbb{E}_{\mu}(\mathrm{wp}\,(Q, f))$$

Inductive Step:

(if)

$$
\begin{aligned}
&\mathbb{E}_{[\![if(B)\,\{\,P\,\}\,else\,\{\,Q\,\}]\!](\mu)}(f) \\
=&\sum_{s\in\mathbb{S}} [\![if(B)\,\{\,P\,\}\,else\,\{\,Q\,\}]\!]\,(\mu)(s)\cdot f(s) \\
&\qquad (\text{PGF } if \text{ semantics}) \\
=&\sum_{s\in\mathbb{S}} \left([\![P]\!]\,(\langle\mu\rangle_B)(s) + [\![Q]\!]\,(\langle\mu\rangle_{\neg B})(s)\right)\cdot f(s) \\
&\qquad (\text{distribute } f(s), \text{ split sum}) \\
=&\sum_{s\in\mathbb{S}} [\![P]\!]\,(\langle\mu\rangle_B)(s)\cdot f(s) + \sum_{s\in\mathbb{S}} [\![Q]\!]\,(\langle\mu\rangle_{\neg B})(s)\cdot f(s) \\
&\qquad (\text{definition expected value}) \\
=&\mathbb{E}_{[\![P]\!](\langle\mu\rangle_B)}(f) + \mathbb{E}_{[\![Q]\!](\langle\mu\rangle_{\neg B})}(f) \\
&\qquad (\text{induction hypothesis}) \\
=&\mathbb{E}_{\langle\mu\rangle_B}(\mathrm{wp}\,(P,f)) + \mathbb{E}_{\langle\mu\rangle_{\neg B}}(\mathrm{wp}\,(Q,f)) \\
=&\sum_{s\in\mathbb{S}} \langle\mu\rangle_B\,(s)\cdot \mathrm{wp}\,(P,f)\,(s) + \sum_{s\in\mathbb{S}} \langle\mu\rangle_{\neg B}\,(s)\cdot \mathrm{wp}\,(Q,f)\,(s) \\
&\qquad (\langle\mu\rangle_B = B\cdot\mu) \\
=&\sum_{s\in\mathbb{S}} [B](s)\cdot\mu(s)\cdot \mathrm{wp}\,(P,f)\,(s) + \sum_{s\in\mathbb{S}} [\neg B](s)\cdot\mu(s)\cdot \mathrm{wp}\,(Q,f)\,(s) \\
&\qquad (\text{merge sums, factor out } \mu(s)) \\
=&\sum_{s\in\mathbb{S}} \mu(s)\cdot\left([B](s)\cdot \mathrm{wp}\,(P,f)\,(s) + [\neg B](s)\cdot \mathrm{wp}\,(Q,f)\,(s)\right) \\
&\qquad (\text{wp } if \text{ semantics}) \\
=&\sum_{s\in\mathbb{S}} \mu(s)\cdot \mathrm{wp}\,(if(B)\,\{\,P\,\}\,else\,\{\,Q\,\},f)\,(s) \\
&\qquad (\text{definition expected value}) \\
=&\mathbb{E}_{\mu}(\mathrm{wp}\,(if(B)\,\{\,P\,\}\,else\,\{\,Q\,\},f))
\end{aligned}
$$

(*concatenation*)

$$
\begin{aligned}
&\mathbb{E}_{[\![P;Q]\!](\mu)}(f) \\
&\qquad \text{(PGF concatenation)} \\
=&\mathbb{E}_{[\![Q]\!]([\![P]\!](\mu))}(f) \\
&\qquad \text{(induction hypothesis)} \\
=&\mathbb{E}_{[\![P]\!](\mu)}(\mathrm{wp}\,(Q, f)) \\
&\qquad \text{(induction hypothesis)} \\
=&\mathbb{E}_{\mu}(\mathrm{wp}\,(P, \mathrm{wp}\,(Q, f))) \\
&\qquad \text{(wp concatenation)} \\
=&\mathbb{E}_{\mu}(\mathrm{wp}\,(P; Q, f))
\end{aligned}
$$

(*prob. choice*)

$$
\begin{aligned}
&\mathbb{E}_{[\![\{P\}[p]\{Q\}]\!](\mu)}(f) \\
=&\sum_{s\in\mathbb{S}} [\![\{P\}[p]\{Q\}]\!]\,(\mu)(s)\cdot f(s) \\
&\qquad \text{(PGF prob. choice)} \\
=&\sum_{s\in\mathbb{S}} p\cdot [\![P]\!]\,(\mu)(s)\cdot f(s) + (1-p)\cdot [\![Q]\!]\,(\mu)(s)\cdot f(s) \\
&\qquad \text{(factor out constants)} \\
=&p\cdot\sum_{s\in\mathbb{S}} [\![P]\!]\,(\mu)(s)\cdot f(s) + (1-p)\cdot\sum_{s\in\mathbb{S}} [\![Q]\!]\,(\mu)(s)\cdot f(s) \\
&\qquad \text{(definition expected value)} \\
=&p\cdot\mathbb{E}_{[\![P]\!](\mu)}(f) + +(1-p)\cdot\mathbb{E}_{[\![Q]\!](\mu)}(f) \\
&\qquad \text{(induction hypothesis)} \\
=&p\cdot\mathbb{E}_{\mu}(\mathrm{wp}\,(P, f)) + (1-p)\cdot\mathbb{E}_{\mu}(\mathrm{wp}\,(Q, f)) \\
=&\mathbb{E}_{\mu}(p\cdot\mathrm{wp}\,(P, f) + (1-p)\cdot\mathrm{wp}\,(Q, f)) \\
=&\mathbb{E}_{\mu}(\mathrm{wp}\,(P[p]Q, f))
\end{aligned}
$$

(*while*) Both, wp and PGF semantics, define the semantics of the while loop using a least fixed point iteration. To prove the equivalence of both semantics, we prove that they are equivalent in every step of the iteration. Recall both iteration functions for the loop body P and the loop condition B:

PGF:

$$\mathrm{H}_{B,P}(f)(G) = \langle G \rangle_{\neg B} + f\left(\mathrm{wh}_{B,P}(G)\right)$$

The iteration starts with the bottom element $\bot$ with $\bot(G) = 0$.

wp: The iteration function has no explicit name in the original paper, but as introduced above, we will call it ww_f, where f is a given expectation. Then,

$$\mathrm{ww}_f(X) = [B] \cdot \mathrm{wp}\left(P, X\right) + [\neg B] \cdot f$$

The iteration using ww_f starts with 0, which is the null expectation.

Proof by induction over n.
Base Case for $n = 0$:

$$\mathbb{E}_{\mathrm{H}^0_{B,P}(\bot)(\mu)}(f) = \mathbb{E}_{\bot(\mu)}(f) = \mathbb{E}_0(f) = 0 = \mathbb{E}_\mu(0) = \mathbb{E}_\mu(\mathrm{ww}^0_f(0))$$

Induction Hypothesis:

$$\mathbb{E}_{\mathrm{H}^n_{B,P}(\bot)(\mu)}(f) = \mathbb{E}_\mu(\mathrm{ww}^n_f(0)) \text{ for some } n \in \mathbb{N}$$

Inductive Step:

$$
\begin{aligned}
&\mathbb{E}_{\mathrm{H}^{n+1}_{B,P}(\bot)(\mu)}(f) \\
=&\sum_{s\in\mathbb{S}} \mathrm{H}^{n+1}_{B,P}(\bot)(\mu)(s)\cdot f(s) \\
&\qquad \text{(split function composition)} \\
=&\sum_{s\in\mathbb{S}} \mathrm{H}_{B,P}(\mathrm{H}^{n}_{B,P}(\bot))(\mu)(s)\cdot f(s) \\
&\qquad \text{(definition of } \mathrm{H}_{B,P}\text{)} \\
=&\sum_{s\in\mathbb{S}} \left(\langle\mu\rangle_{\neg B} + \mathrm{H}^{n}_{B,P}(\bot)(\mathrm{wh}_{B,P}(\mu))\right)(s)\cdot f(s) \\
&\qquad \text{(distribute } f(s)\text{, split sums)} \\
=&\sum_{s\in\mathbb{S}} \langle\mu\rangle_{\neg B}(s)\cdot f(s) + \sum_{s\in\mathbb{S}} \mathrm{H}^{n}_{B,P}(\bot)(\mathrm{wh}_{B,P}(\mu))(s)\cdot f(s) \\
=&\mathbb{E}_{\langle\mu\rangle_{\neg B}}(f) + \mathbb{E}_{\mathrm{H}^{n}_{B,P}(\bot)(\mathrm{wh}_{B,P}(\mu))}(f) \\
&\qquad \text{(inner induction hypothesis)} \\
=&\mathbb{E}_{\langle\mu\rangle_{\neg B}}(f) + \mathbb{E}_{\mathrm{wh}_{B,P}(\mu)}\left(\mathrm{ww}^{n}_{f}(0)\right) \\
&\qquad \text{(definition wh}_{B,P}\text{)} \\
=&\mathbb{E}_{\langle\mu\rangle_{\neg B}}(f) + \mathbb{E}_{[\![P]\!](\langle\mu\rangle_{B})}\left(\mathrm{ww}^{n}_{f}(0)\right) \\
&\qquad \text{(outer induction hypothesis)} \\
=&\mathbb{E}_{\langle\mu\rangle_{\neg B}}(f) + \mathbb{E}_{\langle\mu\rangle_{B}}\left(\mathrm{wp}\left(P, \mathrm{ww}^{n}_{f}(0)\right)\right) \\
=&\sum_{s\in\mathbb{S}} \langle\mu\rangle_{\neg B}(s)\cdot f(s) + \langle\mu\rangle_{B}(s)\cdot \mathrm{wp}\left(P, \mathrm{ww}^{n}_{f}(0)\right)(s) \\
&\qquad (\langle\mu\rangle_{B} = B\cdot\mu) \\
=&\sum_{s\in\mathbb{S}} [\neg B](s)\cdot\mu(s)\cdot f(s) + [B](s)\cdot\mu(s)\cdot \mathrm{wp}\left(P, \mathrm{ww}^{n}_{f}(0)\right)(s) \\
=&\sum_{s\in\mathbb{S}} \mu(s)\cdot\left([\neg B](s)\cdot f(s) + [B](s)\cdot \mathrm{wp}\left(P, \mathrm{ww}^{n}_{f}(0)\right)(s)\right) \\
&\qquad \text{(definition of wp)} \\
=&\sum_{s\in\mathbb{S}} \mu(s)\cdot \mathrm{ww}^{n+1}_{f}(0)(s) \\
=&\mathbb{E}_{\mu}\left(\mathrm{ww}^{n+1}_{f}(0)\right)
\end{aligned}
$$

□

7 Conclusion and Future Work

In this bachelor thesis, we developed a PGF semantics for probabilistic programs with integer program variable valuations. The semantics is based on semantic tuples which are tuples where each entry is a PGFs. The number of entries grows exponentially in the number of program variables. This means that for more than three or four program variables the tuples are hard to handle by hand because they have many entries. However, with some knowledge about the program we can reduce the number of entries that have to be calculated to find a complete semantic description. Another way to reduce complexity is to use PGFs with extended range. Every of these PGFs corresponds to a semantic tuple and can therefore be used as well. We successfully applied the new semantics to a variety of programs and could even extend the results of Gretz et al. concerning two equivalent programs. We showed that the programs are not only equal in expectation but indeed equal in their semantics. In the next step, we found a novel approach to prove the semantics of a loop. Originally, the loop semantics is defined as the sum of all possible loop unrollings. Now, we can also use binary relations called bisimulations. A bisimulation is a set of pairs of PGFs. Each pair is a pair of input and output of the loop that is treated. There are two types of bisimulations. Weak bisimulations allow to prove an overapproximation for the loop's semantics. Strong bisimulations can be used to prove that a pair in the bisimulation is in fact the input and output of the loop. Last, we showed that our PGF semantics is equivalent in expectation to the weakest preexpectation semantics introduced by McIver and Morgan. We proved that the execution of a program in either semantics has the same expected value for an arbitrary property expressed as a function over the program variables.

In this thesis, we proved that two different programs are equal in their semantics. Instead of showing equivalence or nonequivalence of two programs, one could find a measure for the similarity of two programs. This can be done by comparing the semantics of these programs. In our case, this would require a distance function for PGFs. Another idea is to find more use cases and other statements that can be derived from weak and strong bisimulations.

8 Appendix

8.1 Vector Space of FPSs with Extended Range

We want to prove that the PGFs with extended range combined with addition and scalar multiplication form a vector space. Recall the definition of vector space.

Definition 8.1 (Vector Space [**?**])**.** *A vector space V over a field F is a non-empty set V together with two functions, $+\colon V \times V \to V$ and $\cdot\colon F \times V \to V$, which satisfy the following properties:*

Let $u, v, w \in V$ and $r, s \in F$ be arbitrarily chosen.

1. $v + w = w + v$
2. $u + (v + w) = (u + v) + w$
3. *There exists* $0 \in V$ *such that* $0 + v = v$.
4. *There exists* $(-v) \in V$ *such that* $v + (-v) = 0$.
5. $(r \cdot s) \cdot v = r \cdot (s \cdot v)$
6. $r \cdot (u + v) = r \cdot u + r \cdot v$
7. $(r + s) \cdot v = r \cdot v + s \cdot v$
8. $1 \cdot v = v$, *where* 1 *is the multiplicative identity of the field* F

For the PGFs with extended range, we have

- $V = \left\{ \sum_{s \in \mathbb{Z}^k} \mu_s X_1^{s_1} \cdots X_k^{s_k} \middle| \mu_s \in \mathbb{R} \right\}$
- $F = \mathbb{R}$

For $v = \sum_{s \in \mathbb{Z}^k} \mu_s X_1^{s_1} \cdots X_k^{s_k}$ and $w = \sum_{s \in \mathbb{Z}^k} \mu'_s X_1^{s_1} \cdots X_k^{s_k}$ and $r \in \mathbb{R}$, the two operations are defined as follows:

- $+\colon V \times V \to V$

  $$v + w = \sum_{s \in \mathbb{Z}^k} (\mu_s + \mu'_s) X_1^{s_1} \cdots X_k^{s_k}$$

- $\cdot\colon \mathbb{R} \times V \to V$

  $$r \cdot v = \sum_{s \in \mathbb{Z}^k} (r \cdot \mu_s) X_1^{s_1} \cdots X_k^{s_k}$$

To prove that $(V, \cdot, +)$ is a vector space, we have to prove properties 1 to 8.

Theorem ?? (continuing from p. **??**)**.** *The set of FPSs with extended range combined with addition and scalar multiplication forms a* vector space.

Proof. Let $u, v, w \in V$ with their coefficients σ_s, μ_s and τ_s, respectively, be arbitrarily chosen. Additionally, let $r, s \in \mathbb{R}$ be arbitrarily chosen.

1. $v + w$

$$= \sum_{s \in \mathbb{Z}^k} (\mu_s + \tau_s) X_1^{s_1} \cdots X_k^{s_k}$$

$$= \sum_{s \in \mathbb{Z}^k} (\tau_s + \mu_s) X_1^{s_1} \cdots X_k^{s_k}$$

$$= w + v$$

2. $u + (v + w)$

$$= \sum_{s \in \mathbb{Z}^k} (\sigma_s + (\mu_s + \tau_s)) X_1^{s_1} \cdots X_k^{s_k}$$

$$= \sum_{s \in \mathbb{Z}^k} ((\sigma_s + \mu_s) + \tau_s) X_1^{s_1} \cdots X_k^{s_k}$$

$$= (u + v) + w$$

3. Let $0 = \sum_{s \in \mathbb{Z}^k} 0 X_1^{s_1} \cdots X_k^{s_k} \in V$. Then,

$$0 + v$$

$$= \sum_{s \in \mathbb{Z}^k} (0 + \mu_s) X_1^{s_1} \cdots X_k^{s_k}$$

$$= \sum_{s \in \mathbb{Z}^k} \mu_s X_1^{s_1} \cdots X_k^{s_k}$$

$$= v$$

4. Let $-v = \sum_{s \in \mathbb{Z}^k} (-\mu_s) X_1^{s_1} \cdots X_k^{s_k} \in V$. Then,

$$v + (-v)$$

$$= \sum_{s \in \mathbb{Z}^k} (\mu_s + (-\mu_s)) X_1^{s_1} \cdots X_k^{s_k}$$

$$= \sum_{s \in \mathbb{Z}^k} 0 X_1^{s_1} \cdots X_k^{s_k}$$

$$= 0$$

5. $(r \cdot s) \cdot v$

$$= \sum_{s \in \mathbb{Z}^k} ((r \cdot s) \cdot \mu_s) X_1^{s_1} \cdots X_k^{s_k}$$

$$= \sum_{s \in \mathbb{Z}^k} (r \cdot (s \cdot \mu_s)) X_1^{s_1} \cdots X_k^{s_k}$$

$$= r \cdot \sum_{s \in \mathbb{Z}^k} (s \cdot \mu_s) X_1^{s_1} \cdots X_k^{s_k}$$

$$= r \cdot (s \cdot v)$$

6. $r \cdot (v + w)$

$$= r \cdot \left(\sum_{s \in \mathbb{Z}^k} (\mu_s + \tau_s) X_1^{s_1} \cdots X_k^{s_k} \right)$$

$$= \sum_{s \in \mathbb{Z}^k} (r \cdot (\mu_s + \tau_s)) X_1^{s_1} \cdots X_k^{s_k}$$

$$= \sum_{s \in \mathbb{Z}^k} (r \cdot \mu_s + r \cdot \tau_s) X_1^{s_1} \cdots X_k^{s_k}$$

$$= \sum_{s \in \mathbb{Z}^k} (r \cdot \mu_s) X_1^{s_1} \cdots X_k^{s_k} + \sum_{s \in \mathbb{Z}^k} (r \cdot \tau_s) X_1^{s_1} \cdots X_k^{s_k}$$

$$= r \cdot \sum_{s \in \mathbb{Z}^k} \mu_s X_1^{s_1} \cdots X_k^{s_k} + r \cdot \sum_{s \in \mathbb{Z}^k} \tau_s X_1^{s_1} \cdots X_k^{s_k}$$

$$= r \cdot v + r \cdot w$$

7. $(r + s) \cdot v$

$$= \sum_{s \in \mathbb{Z}^k} ((r + s) \cdot \mu_s) X_1^{s_1} \cdots X_k^{s_k}$$

$$= \sum_{s \in \mathbb{Z}^k} (r \cdot \mu_s + s \cdot \mu_s) X_1^{s_1} \cdots X_k^{s_k}$$

$$= \sum_{s \in \mathbb{Z}^k} (r \cdot \mu_s) X_1^{s_1} \cdots X_k^{s_k} + \sum_{s \in \mathbb{Z}^k} (s \cdot \mu_s) X_1^{s_1} \cdots X_k^{s_k}$$

$$= r \cdot \sum_{s \in \mathbb{Z}^k} \mu_s X_1^{s_1} \cdots X_k^{s_k} + s \cdot \sum_{s \in \mathbb{Z}^k} \mu_s X_1^{s_1} \cdots X_k^{s_k}$$

$$= r \cdot v + s \cdot v$$

8. $1 \cdot v$

$$= \sum_{s \in \mathbb{Z}^k} (1 \cdot \mu_s) X_1^{s_1} \cdots X_k^{s_k}$$

$$= \sum_{s \in \mathbb{Z}^k} \mu_s X_1^{s_1} \cdots X_k^{s_k}$$

$$= v$$

$(V, \cdot, +)$ satisfies all 8 properties, so it is a vector space. □

8.2 D and L Are ω-Complete Partial Orders

D_k was defined as the set of all PGFs over k variables with its coefficients and absolute value in $[0, 1]$. L_k is the set of all functions mapping D_k to D_k. To show that D_k and L_k are ω-complete partial orders [**?**], we have to show the following:

1. D_k and L_k are reflexive.
2. D_k and L_k are transitive.
3. D_k and L_k are antisymmetric.
4. Every ω-chain in D_k or L_k has a supremum.

Lemma ?? (continuing from p. **??**)**.** *D_k is an ω-complete partial order for all $k \in \mathbb{N} \setminus \{0\}$.*

Proof. In the following, let r, s and t be arbitrary elements in D_k with their coefficients ρ_i, σ_i and τ_i for $i \in \mathbb{Z}^k$.

1. $\forall i \in \mathbb{Z}^k \colon \rho_i \leq \rho_i \iff r \sqsubseteq_D r$
2. Assume $r \sqsubseteq_D s$ and $s \sqsubseteq_D t$. Then

$$\begin{aligned} & \forall i \in \mathbb{Z}^k \colon \rho_i \leq \sigma_i \wedge \forall i \in \mathbb{Z}^k \colon \sigma_i \leq \tau_i \\ \implies & \forall i \in \mathbb{Z}^k \colon \rho_i \leq \sigma_i \wedge \sigma_i \leq \tau_i \\ \implies & \forall i \in \mathbb{Z}^k \colon \rho_i \leq \tau_i \\ \implies & r \sqsubseteq_D t \end{aligned}$$

3. Assume $r \sqsubseteq_D s$ and $s \sqsubseteq_D r$. Then

$$\begin{aligned} & \forall i \in \mathbb{Z}^k \colon \rho_i \leq \sigma_i \wedge \forall i \in \mathbb{Z}^k \colon \sigma_i \leq \rho_i \\ \implies & \forall i \in \mathbb{Z}^k \colon \rho_i \leq \sigma_i \wedge \sigma_i \leq \rho_i \\ \implies & \forall i \in \mathbb{Z}^k \colon \sigma_i = \rho_i \\ \implies & r = t \end{aligned}$$

4. Note: Because of the various subscripts, only in this fourth item, we will denote the coefficient of a PGF G corresponding to $i \in \mathbb{Z}^k$ with $G(i)$.
   Let $d_1 \sqsubseteq_D d_2 \sqsubseteq_D d_3 \sqsubseteq_D \ldots$ be an ω-chain in D_k. The coefficients of d_n are now $d_n(i)$ for $i \in \mathbb{Z}^k$. The supremum $d_{\sup}$ of the ω-chain is given by

$$d_{\sup} = \sup_{n\in\mathbb{N}}\{d_n\} = \sum_{i\in\mathbb{Z}^k} \sup_{n\in\mathbb{N}} \{d_n(i)\}\, X_1^{i_1} \cdots X_k^{i_k}$$

   The supremum $\sup_{n\in\mathbb{N}}\{d_n(i)\}$ exists because $d_n(i) \leq 1$ for all $i \in \mathbb{Z}^k$ by definition of D_k. It remains to prove that $d_{\sup}$ is in fact the supremum of the ω-chain.

   i) *$d_{\sup}$ is an upper bound for all d_n.*
   Assume the contrary. Then there is an index $n \in \mathbb{N}$ such that $d_{\sup} \sqsubsetneq_D d_n$. Then there exists an $i \in \mathbb{Z}^k$ such that $d_n(i) > d_{\sup}(i) = \sup_{n\in\mathbb{N}}\{d_n(i)\}$. This is a contradiction to the definition of supremum, so $d_{\sup}$ is an upper bound for all d_n.

   ii) *There is no d' with $d_n \sqsubseteq_D d' \sqsubsetneq_D d_{\sup}$ for all $n \in \mathbb{N}$.*
   Assume d' exists. Then there exists $i \in \mathbb{Z}$ such that $d_n(i) \leq d'(i) < d_{\sup}(i) = \sup_{n\in\mathbb{N}}\{d_n(i)\}$. This is a contradiction to the definition of supremum, so d' cannot exist.

   By the items i) and ii) above, $d_{\sup}$ is the supremum of the ω-chain.

All four items above are proven, so D_k is an ω-complete partial order. □

Lemma 8.1. *L_k is an ω-complete partial order for all $\mathbb{N} \setminus \{0\}$.*

Proof. In the following, let f, g and h be arbitrary elements in L_k.

1. D_k is reflexive, so $\forall d \in D_k\colon f(d) \sqsubseteq_D f(d) \iff f \sqsubseteq_L f$.

2. Assume $f \sqsubseteq_L g$ and $g \sqsubseteq_L h$. Then

$$\begin{aligned}
&\forall d \in D_k\colon f(d) \sqsubseteq_D g(d) \wedge \forall d \in D_k\colon g(d) \sqsubseteq_D h(d) \\
\Longrightarrow\ &\forall d \in D_k\colon f(d) \sqsubseteq_D g(d) \wedge g(d) \sqsubseteq_D h(d) \\
&\qquad (D_k \text{ is transitive}) \\
\Longrightarrow\ &\forall d \in D_k\colon f(d) \sqsubseteq_D h(d) \\
&\qquad (\text{definition of } \sqsubseteq_L) \\
\Longrightarrow\ &f \sqsubseteq_L h
\end{aligned}$$

3. Assume $f \sqsubseteq_L g$ and $g \sqsubseteq_L f$. Then

$$\begin{aligned}
&\forall d \in D_k\colon f(d) \sqsubseteq_D g(d) \wedge \forall d \in D_k\colon g(d) \sqsubseteq_D f(d) \\
\Longrightarrow\ &\forall d \in D_k\colon f(d) \sqsubseteq_D g(d) \wedge g(d) \sqsubseteq_D f(d) \\
&\qquad (D_k \text{ is antisymmetric}) \\
\Longrightarrow\ &\forall d \in D_k\colon f(d) = g(d) \\
\Longrightarrow\ &f = g
\end{aligned}$$

4. Let $f_1 \sqsubseteq_L f_2 \sqsubseteq_L f_3 \sqsubseteq_L \ldots$ be an ω-chain in L_k. The supremum $l_{\sup}$ of the ω-chain is given by

$$l_{\sup}(d) = \sup_{n\in\mathbb{N}}\{f_n(d)\}$$

The supremum on the the right hand side exists because $f_1(d) \sqsubseteq_D f_2(d) \sqsubseteq_D f_3(d) \sqsubseteq_D \ldots$ is an ω-chain in D_k for every $d \in D_k$. It remains to prove that $l_{\sup}$ is in fact the supremum of the ω-chain.

   i) *$f_{\sup}$ is an upper bound for all f_n.*
   Assume the contrary. Then there is an index $n \in \mathbb{N}$ such that $f_{\sup} \sqsubsetneq_L f_n$. Then there exists a $d \in D_k$ such that $f_n(d) \sqsupsetneq_L f_{\sup}(d) = \sup_{n\in\mathbb{N}}\{f_n(d)\}$. This is a contradiction to the definition of supremum, so $f_{\sup}$ is an upper bound for all f_n.

   ii) *There is no f' with $f_n \sqsubseteq_L f' \sqsubsetneq_L f_{\sup}$ for all $n \in \mathbb{N}$.*
   Assume f' exists. Then there exists $d \in D_k$ such that $f_n(d) \sqsubseteq_D f'(d) \sqsubsetneq_D f_{\sup}(d) = \sup_{n\in\mathbb{N}}\{f_n(d)\}$. This is a contradiction to the definition of supremum, so f' cannot exist.

   By the items i) and ii) above, $f_{\sup}$ is the supremum of the ω-chain.

All four items above are proven, so L_k is an ω-complete partial order. □

8.3 Scott-continuity of $\mathrm{H}_{B,P}$

Lemma ?? (continuing from p. **??**)**.** *Let B be a Boolean condition and P be a program. Then*

$$\mathrm{H}_{B,P}(f)(G) = \langle G\rangle_{\neg B} + f\left(\mathrm{wh}_{B,P}(G)\right) \text{ is Scott-continuous.}$$

Proof. Let $f_1 \sqsubseteq_L f_2 \sqsubseteq_L f_3 \sqsubseteq_L \ldots$ be an ω-chain in L. To show the Scott-continuity of $\mathrm{H}_{B,P}$, it suffices to prove that

$$\mathrm{H}_{B,P}\left(\sup_{n\in\mathbb{N}}\{f_n\}\right) = \sup_{n\in\mathbb{N}}\{\mathrm{H}_{B,P}(f_i)\}$$

Starting with the left hand side of the equation, we complete the proof.

$$\begin{aligned}
&\mathrm{H}_{B,P}\left(\sup_{n\in\mathbb{N}}\{f_n\}\right)\\
=&\lambda G.\,\langle G\rangle_{\neg B} + \left(\sup_{n\in\mathbb{N}}\{f_n\}\right)\left(\llbracket P\rrbracket\left(\langle G\rangle_B\right)\right)\\
=&\lambda G.\,\langle G\rangle_{\neg B} + \sup_{n\in\mathbb{N}}\{f_n(\llbracket P\rrbracket\left(\langle G\rangle_B\right))\}\\
=&\sup_{n\in\mathbb{N}}\{\lambda G.\,\langle G\rangle_{\neg B} + f_n\left(\llbracket P\rrbracket\left(\langle G\rangle_B\right)\right)\}\\
=&\sup_{n\in\mathbb{N}}\{\mathrm{H}_{B,P}(f_n)\}
\end{aligned}$$

□

8.4 Closed Form of $\mathrm{H}^n_{B,P}(\bot)$

We want to prove a closed form for $\mathrm{H}^n_{B,P}(\bot)(G)$.

Lemma ?? (continuing from p. **??**)**.** *Proof.* Proof by induction over n.
Base case for $n = 0$:

$$\mathrm{H}^0_{B,P}(\bot)(G) = \bot(G) = 0 = \sum_{i=0}^{0-1} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B}$$

Note the empty sum in the last step.
Induction hypothesis:

$$\mathrm{H}^n_{B,P}(\bot)(G) = \sum_{i=0}^{n-1} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B}, \text{ for some } n \in \mathbb{N}$$

Inductive step:

$$\begin{aligned}
&\mathrm{H}^{n+1}_{B,P}(\bot)(G) \\
=&\, \mathrm{H}_{B,P}(\mathrm{H}^n_{B,P}(\bot))(G) \\
&\qquad \text{(definition of } \mathrm{H}_{B,P}\text{)} \\
=&\, \langle G \rangle_{\neg B} + \mathrm{H}^n_{B,P}(\bot)(\llbracket P \rrbracket (\langle G \rangle_B)) \\
&\qquad \text{(definition of } \mathrm{wh}_{B,P}\text{)} \\
=&\, \langle G \rangle_{\neg B} + \mathrm{H}^n_{B,P}(\bot)(\mathrm{wh}_{B,P}(G)) \\
&\qquad \text{(apply induction hypothesis)} \\
=&\, \langle G \rangle_{\neg B} + \sum_{i=0}^{n-1} \left\langle \mathrm{wh}^i_{B,P}(\mathrm{wh}_{B,P}(G)) \right\rangle_{\neg B} \\
=&\, \langle G \rangle_{\neg B} + \sum_{i=0}^{n-1} \left\langle \mathrm{wh}^{i+1}_{B,P}(G) \right\rangle_{\neg B} \\
&\qquad \text{(offset indices)} \\
=&\, \langle G \rangle_{\neg B} + \sum_{i=1}^{n} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \\
&\qquad \left(\mathrm{wh}^0_{B,P} \text{ is the identity function}\right) \\
&\left\langle \mathrm{wh}^0_{B,P}(G) \right\rangle_{\neg B} + \sum_{i=1}^{n} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \\
=&\sum_{i=0}^{n} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B}
\end{aligned}$$

□

8.5 A Series As Loop Semantics

Theorem ?? (continuing from p. **??**)**.** *Let B be a Boolean condition and P a program. Then*

$$[\![\mathit{while}\,(B)\{P\}]\!]\,(G) = \sum_{i=0}^{\infty} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B}$$

Proof. We know that

$$\sum_{i=0}^{\infty} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} = \lim_{n\in\mathbb{N}} \sum_{i=0}^{n} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B}$$

Since the partial sums are monotonically increasing, we can replace lim with sup.

$$\begin{aligned}
&\sum_{i=0}^{\infty} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \\
=&\sup_{n\in\mathbb{N}} \left\{ \sum_{i=0}^{n} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \right\} \\
=&\sup_{n\in\mathbb{N}} \left\{ \mathrm{H}^{n+1}_{B,P}(\bot)(G) \right\} \\
&\qquad \left(\mathrm{H}^0_{B,P}(\bot)(G) = 0\right) \\
=&\sup_{n\in\mathbb{N}} \left\{ \mathrm{H}^{n}_{B,P}(\bot)(G) \right\} \\
&\qquad (\mathrm{H}_{B,P} \text{ is Scott-continuous}) \\
=&\sup_{n\in\mathbb{N}} \left\{ \mathrm{H}^{n}_{B,P}(\bot) \right\}(G) \\
=&\,[\![\mathit{while}\,(B)\{P\}]\!]\,(G)
\end{aligned}$$

□

8.6 Canonical Example

In this proof, we will not show the exact statement for the canonical example, but a more general one. Therefore, we slightly modify the loop body P by parametrising it with a variable $p \in [0,1]$.

$$P_p = \{X := X + 1\}[p]\{F := 1\}$$

The statement needed for the canonical example then is the special case $p = \frac{1}{2}$.

Lemma 8.2. *Let $B = (F = 0)$ be a Boolean condition and P_p as above with $p \in [0,1]$. Then*

$$\left\langle \mathrm{wh}^i_{B,P_p}(1) \right\rangle_{\neg B} = (1-p)\cdot p^{i-1}X^{i-1}F^1, \text{ where } i > 0$$

Proof. We will prove the equation above without the restriction to $\neg B$ and apply it afterwards. Proof by induction over i.
Base Case for $i = 1$:

$$\mathrm{wh}^1_{B,P_p}(1) = p^1X^1F^0 + (1-p)^1X^0F^1$$

Induction Hypothesis:

$$\mathrm{wh}^{i}_{B,P_p}(1) = p^i X^i F^0 + (1-p)\cdot p^{i-1} X^{i-1} F^1 \text{ for some } i > 0$$

Inductive Step:

$$\begin{aligned}
&\mathrm{wh}^{i+1}_{B,P_p}(1)\\
=&\,\mathrm{wh}_{B,P_p}\left(\mathrm{wh}^{i}_{B,P_p}(1)\right)\\
&\qquad\left(\text{definition of } \mathrm{wh}_{B,P_p}\right)\\
=&\,[\![P]\!]\left(\left\langle \mathrm{wh}^{i}_{B,P_p}(1)\right\rangle_B\right)\\
&\qquad(\text{induction hypothesis})\\
=&\,[\![P]\!]\left(\left\langle p^i X^i F^0 + (1-p)\cdot p^{i-1}X^{i-1}F^1\right\rangle_B\right)\\
&\qquad(B = (F=0))\\
=&\,[\![P]\!]\left(p^i X^i F^0\right)\\
&\qquad(\text{semantics of } P)\\
=&\,p\cdot p^i X^{i+1} F^0 + (1-p)\cdot p^i X^i F^1\\
=&\,p^{i+1} X^{i+1} F^0 + (1-p)\cdot p^i X^i F^1
\end{aligned}$$

The intermediate claim is proven by induction. The final result is now obtained by restriction to $\neg B$:

$$\left\langle \mathrm{wh}^{i}_{B,P_p}(1)\right\rangle_{\neg B} = \left\langle p^i X^i F^0 + (1-p)\cdot p^{i-1}X^{i-1}F^1\right\rangle_{\neg B} = (1-p)\cdot p^{i-1}X^{i-1}F^1 \qquad \square$$

Summing all $\mathrm{wh}^{i}_{B,P}(1)$ gives us the semantics of the more general loop:

$$\begin{aligned}
&[\![\mathit{while}\,(B)\{P_p\}]\!]\,(1)\\
=&\left\langle \mathrm{wh}^{0}_{B,P_p}(1)\right\rangle_{\neg B} + \sum_{i=1}^{\infty}\left\langle \mathrm{wh}^{i}_{B,P_p}(1)\right\rangle_{\neg B}\\
=&\,0 + \sum_{i=1}^{\infty}(1-p)\cdot p^{i-1}X^{i-1}F^1\\
&\qquad(\text{factor out, transform indices})\\
=&\,(1-p)\cdot\sum_{i=0}^{\infty} p^i X^i F^1\\
&\qquad(\text{closed form of geometric series})\\
=&\,\frac{(1-p)F}{1-pX}
\end{aligned}$$

8.7 Example with Termination Probability < 1

Lemma 8.3. *For the example with termination probability* < 1*,*

$$\left\langle \mathrm{wh}^i_{B,P}(1) \right\rangle_{\neg B} = \frac{1}{2} \cdot \left(\frac{1}{4}\right)^{i-1} X^{i-1}F^1, \text{ where } i > 0$$

Proof. We will prove the equation above without the restriction to $\neg B$ and apply it afterwards. Proof by induction over i.
Base Case for $i = 1$:

$$\mathrm{wh}^1_{B,P}(1) = \frac{1}{4}X^1F^0 + \frac{1}{2}X^0F^1$$

Induction Hypothesis:

$$\mathrm{wh}^i_{B,P}(1) = \left(\frac{1}{4}\right)^i X^iF^0 + \frac{1}{2} \cdot \left(\frac{1}{4}\right)^{i-1} X^{i-1}F^1 \text{ for some } i > 0$$

Inductive Step:

$$\begin{aligned}
&\mathrm{wh}^{i+1}_{B,P}(1) \\
=&\,\mathrm{wh}_{B,P}(\mathrm{wh}^i_{B,P}(1)) \\
&\qquad (\text{definition of } \mathrm{wh}_{B,P}) \\
=&\,[\![P]\!]\left(\left\langle \mathrm{wh}^i_{B,P}(1) \right\rangle_B\right) \\
&\qquad (\text{induction hypothesis}) \\
=&\,[\![P]\!]\left(\left\langle \left(\frac{1}{4}\right)^i X^iF^0 + \frac{1}{2} \cdot \left(\frac{1}{4}\right)^{i-1} X^{i-1}F^1 \right\rangle_B\right) \\
&\qquad (B = (F = 0)) \\
=&\,[\![P]\!]\left(\left(\frac{1}{4}\right)^i X^iF^0\right) \\
&\qquad (P = \{\{X := X + 1\}[0.5]\{\mathit{abort}\}\}\,[0.5]\{F := 1\}) \\
=&\frac{1}{4} \cdot \left(\frac{1}{4}\right)^i X^{i+1}F^0 + \frac{1}{2} \cdot \left(\frac{1}{4}\right)^i X^iF^1 \\
=&\left(\frac{1}{4}\right)^{i+1} X^{i+1}F^0 + \frac{1}{2} \cdot \left(\frac{1}{4}\right)^i X^iF^1
\end{aligned}$$

The intermediate claim is proven by induction. The final result is now obtained by restriction to $\neg B$.

$$\left\langle \mathrm{wh}^i_{B,P}(1) \right\rangle_{\neg B} = \left\langle \left(\frac{1}{4}\right)^i X^iF^0 + \frac{1}{2} \cdot \left(\frac{1}{4}\right)^{i-1} X^{i-1}F^1 \right\rangle_{\neg B} = \frac{1}{2} \cdot \left(\frac{1}{4}\right)^{i-1} X^{i-1}F^1 \qquad \square$$

8.8 Equivalent Programs

Consecutive Loops

Two statements about the loop L^- in the first program of Figure **??** remain to be proved. First, we have to show the general form of $\mathrm{wh}^i_{B,P}$ and then the simplification of the infinite sum of all these terms. L^- operates on the PGF generated from L^+, the first loop in the program. After L^+ and before L^-, the variable F is set to 0, which only changes F's exponent in the previous PGF. The input for L^- then is the following geometric distribution G, as proven in **??**.

$$G = [\![F := 0; L^+]\!]\,(1,0) = (1-p)\cdot\left(\sum_{i=0}^{\infty} p^i X^i F^0, 0\right)$$

Lemma 8.4. *For the loop L^-, the following holds:*

$$\begin{aligned}&\left\langle \mathrm{wh}^n_{B,P}\,(G)\right\rangle_{\neg B}\\ =&(1-q)\cdot q^{n-1}\cdot(1-p)\cdot\left(\sum_{i=0}^{\infty} p^{i+(n-1)}X^iF^1, \sum_{i=1}^{n-1} p^{(n-1)-i}X^{-i}F^1\right)\end{aligned}$$

Proof. We will prove the equation above without the restriction to $\neg B$ and apply it afterwards. Proof by induction over n.
Base Case for $n = 1$:

$$\begin{aligned}&\mathrm{wh}^1_{B,P}(G)\\ =&q\cdot[\![X := X-1]\!]\,(G) + (1-q)\cdot[\![F := 1]\!]\,(G)\\ =&q\cdot(1-p)\cdot\left(\sum_{i=0}^{\infty} p^{i+1}X^iF^0, 1X^{-1}F^0\right)\\ &+(1-q)\cdot(1-p)\cdot\left(\sum_{i=0}^{\infty} p^iX^iF^1, 0\right)\\ &\qquad\text{(empty sum in second entry is zero)}\\ =&q\cdot(1-p)\cdot\left(\sum_{i=0}^{\infty} p^{i+1}X^iF^0, 1X^{-1}F^0\right)\\ &+(1-q)\cdot(1-p)\cdot\left(\sum_{i=0}^{\infty} p^iX^iF^1, \sum_{i=1}^{1-1} p^{(1-1)-i}X^{-i}F^q\right)\end{aligned}$$

Induction Hypothesis:

For some $n > 0$:

$$\begin{aligned}
&\mathrm{wh}^n_{B,P}(G) \\
=&q^n \cdot (1-p) \cdot \left(\sum_{i=0}^{\infty} p^{i+n} X^i F^0, \sum_{i=1}^{n} p^{n-i} X^{-i} F^0 \right) \\
&+ (1-q) \cdot q^{n-1} \cdot (1-p) \cdot \left(\sum_{i=0}^{\infty} p^{i+(n-1)} X^i F^1, \sum_{i=1}^{n-1} p^{(n-1)-i} X^{-i} F^1 \right)
\end{aligned}$$

Inductive Step: One can easily verify that restricting $\mathrm{wh}^n_{B,P}(G)$ from the induction hypothesis to B gives the first summand of $\mathrm{wh}^n_{B,P}(G)$, because in the second summand, F always has value 1.

$$\left\langle \mathrm{wh}^n_{B,P}(G) \right\rangle_B = q^n \cdot (1-p) \cdot \left(\sum_{i=0}^{\infty} p^{i+n} X^i F^0, \sum_{i=1}^{n} p^{n-i} X^{-i} F^0 \right)$$

We can use this for the inductive step.

$$
\begin{aligned}
&\mathrm{wh}_{B,P}^{n+1}(G)\\
=&\,\mathrm{wh}_{B,P}\left(\mathrm{wh}_{B,P}^{n}(G)\right)\\
&\qquad\text{(definition of } \mathrm{wh}_{B,P}\text{)}\\
=&\,[\![P]\!]\left(\left\langle \mathrm{wh}_{B,P}^{n}(G)\right\rangle_{B}\right)\\
&\qquad\text{(induction hypothesis + equation above)}\\
=&\,[\![P]\!]\left(q^n\cdot(1-p)\cdot\left(\sum_{i=0}^{\infty}p^{i+n}X^iF^0,\sum_{i=1}^{n}p^{n-i}X^{-i}F^0\right)\right)\\
&\qquad\text{(apply } P\text{)}\\
=&\,q\cdot q^n\cdot(1-p)\cdot\left(\sum_{i=0}^{\infty}p^{i+n+1}X^iF^0,p^nX^{-1}F^0+\sum_{i=1}^{n}p^{n-i}X^{-i-1}F^0\right)\\
&+(1-q)\cdot q^n\cdot(1-p)\cdot\left(\sum_{i=0}^{\infty}p^{i+n}X^iF^1,\sum_{i=1}^{n}p^{n-i}X^{-i}F^1\right)\\
&\qquad\text{(transform sum indices in second entry)}\\
=&\,q^{n+1}\cdot(1-p)\cdot\left(\sum_{i=0}^{\infty}p^{i+(n+1)}X^iF^0,p^nX^{-1}F^0+\sum_{i=2}^{n+1}p^{(n+1)-i}X^{-i}F^0\right)\\
&+(1-q)\cdot q^n\cdot(1-p)\cdot\left(\sum_{i=0}^{\infty}p^{i+n}X^iF^1,\sum_{i=1}^{n}p^{n-i}X^{-i}F^1\right)\\
&\qquad\text{(merge sum in second entry)}\\
=&\,q^{n+1}\cdot(1-p)\cdot\left(\sum_{i=0}^{\infty}p^{i+(n+1)}X^iF^0,\sum_{i=1}^{n+1}p^{(n+1)-i}X^{-i}F^0\right)\\
&+(1-q)\cdot q^{(n+1)-1}\cdot(1-p)\cdot\left(\sum_{i=0}^{\infty}p^{i+((n+1)-1)}X^iF^1,\sum_{i=1}^{n}p^{((n+1)-1)-i}X^{-i}F^1\right)
\end{aligned}
$$

The induction hypothesis is proven by induction. The final result is now obtained by restriction to $\neg B$. One can easily see that restricting $\mathrm{wh}_{B,P}^{n}(G)$ to $\neg B$ leaves only the second summand, because F has value 1 only in this part of the PGF.

$$
\begin{aligned}
&\left\langle \mathrm{wh}_{B,P}^{n}(G)\right\rangle_{\neg B}\\
=&(1-q)\cdot q^{n-1}\cdot(1-p)\cdot\left(\sum_{i=0}^{\infty}p^{i+(n-1)}X^iF^1,\sum_{i=1}^{n-1}p^{(n-1)-i}X^{-i}F^1\right)
\end{aligned}
$$

□

Next we want to prove the whole semantics of L^-.

Lemma 8.5. *The following holds.*

$$\begin{aligned}&[\![L^-]\!]\left((1-p)\cdot\left(\sum_{i=0}^{\infty}p^iX^iF^0,0\right)\right)\\=&\frac{(1-q)\cdot(1-p)}{1-qp}\cdot\left(\sum_{n=0}^{\infty}p^nX^nF^1,q\cdot\sum_{n=0}^{\infty}q^nX^{-(n+1)}F^1\right)\end{aligned}$$

Proof.

$$\begin{aligned}&[\![L^-]\!]\left((1-p)\cdot\left(\sum_{i=0}^{\infty}p^iX^iF^0,0\right)\right)\\&\qquad\text{(loop semantics)}\\=&\sum_{n=0}^{\infty}\left\langle\mathrm{wh}^n_{B,P}\left((1-p)\cdot\left(\sum_{i=0}^{\infty}p^iX^iF^0,0\right)\right)\right\rangle_{\neg B}\\&\qquad\text{(insert result of previous proof)}\\=&\sum_{n=1}^{\infty}(1-q)\cdot q^{n-1}\cdot(1-p)\cdot\left(\sum_{i=0}^{\infty}p^{i+(n-1)}X^iF^1,\sum_{i=1}^{n-1}p^{(n-1)-i}X^{-i}F^1\right)\\&\qquad\text{(factor out)}\\=&(1-q)\cdot(1-p)\cdot\sum_{n=1}^{\infty}q^{n-1}\cdot\left(\sum_{i=0}^{\infty}p^{i+(n-1)}X^iF^1,\sum_{i=1}^{n-1}p^{(n-1)-i}X^{-i}F^1\right)\\&\qquad\text{(expand sums)}\\=&(1-q)\cdot(1-p)\cdot\Big(\\&\quad q^0\cdot\left(p^0X^0F^1+p^1X^1F^1+\ldots,0\right)&&\mid n=1\\&+q^1\cdot\left(p^1X^0F^1+p^2X^1F^1+\ldots,p^0X^{-1}F^1\right)&&\mid n=2\\&+q^2\cdot\left(p^2X^0F^1+p^3X^1F^1+\ldots,p^1X^{-1}F^1+p^0X^{-2}F^1\right)&&\mid n=3\\&+q^3\cdot\left(p^3X^0F^1+p^4X^1F^1+\ldots,p^2X^{-1}F^1+p^1X^{-2}F^1+p^0X^{-3}F^1\right)&&\mid n=4\\&+\ldots\Big)\end{aligned}$$

If we look at the entries in the last expression columnwise, the factors of X^nF^1 for $n\in\mathbb{Z}$ are right under each other. Together with the factor q^n in front, they form the geometric series $\sum_{i=0}^{\infty}q^ip^i$ multiplied by a constant factor depending on n. Hence, the sum of all X^nF^1

simplifies to a single expression as follows:

$$p^n \cdot \left(\sum_{i=0}^{\infty} q^i p^i \right) \cdot X^n F^1, \text{ where } n \geq 0$$

$$q^n \cdot \left(\sum_{i=0}^{\infty} q^i p^i \right) \cdot X^{-n} F^1, \text{ where } n < 0$$

Continuing the equations from above, this gives:

$$= (1-q) \cdot (1-p) \cdot \left(\sum_{n=0}^{\infty} p^n \cdot \left(\sum_{i=0}^{\infty} q^i p^i \right) \cdot X^n F^1, \sum_{n=1}^{\infty} q^n \cdot \left(\sum_{i=0}^{\infty} q^i p^i \right) \cdot X^{-n} F^1 \right)$$

(geometric series)

$$= (1-q) \cdot (1-p) \cdot \left(\sum_{n=0}^{\infty} p^n \cdot \frac{1}{1-qp} \cdot X^n F^1, \sum_{n=1}^{\infty} q^n \cdot \frac{1}{1-qp} \cdot X^{-n} F^1 \right)$$

(factor out)

$$= \frac{(1-q) \cdot (1-p)}{1-qp} \cdot \left(\sum_{n=0}^{\infty} p^n X^n F^1, \sum_{n=1}^{\infty} q^n X^{-n} F^1 \right)$$

(transform indices in second entry)

$$= \frac{(1-q) \cdot (1-p)}{1-qp} \cdot \left(\sum_{n=0}^{\infty} p^n X^n F^1, q \cdot \sum_{n=0}^{\infty} q^n X^{-(n+1)} F^1 \right)$$

□

Separated Loops

For the loop L^- in the second program in Figure **??**, we have to prove a statement about $\left\langle \mathrm{wh}^i_{B,P}(1,0) \right\rangle_{\neg B}$

Lemma 8.6. *For the loop L^-, the following holds:*

$$\left\langle \mathrm{wh}^i_{B,P}(1,0) \right\rangle_{\neg B} = (1-q) \cdot \left(0, q^{i-1} X^{-i} F^1\right)$$

Proof. We will prove the equation above without the restriction to $\neg B$ and apply it afterwards. Proof by induction over i.
Base Case for $i = 1$:

$$\mathrm{wh}^i_{B,P}(1,0) = \left(0, qX^{-1}F^0\right) + \left(0, (1-q)X^{-1}F^1\right)$$

Induction Hypothesis:

$$\mathrm{wh}^i_{B,P}(1,0) = \left(0, q^i X^{-i} F^0\right) + \left(0, (1-q) \cdot q^{i-1} X^{-i} F^1\right) \text{ for some } i > 0$$

Inductive Step:

$$\begin{aligned}
&\mathrm{wh}_{B,P}^{i+1}(1,0)\\
=&\,\mathrm{wh}_{B,P}\left(\mathrm{wh}_{B,P}^{i}(1,0)\right)\\
&\qquad(\text{definition of } \mathrm{wh}_{B,P})\\
=&\,[\![P]\!]\left(\left\langle \mathrm{wh}_{B,P}^{i}(1,0)\right\rangle_{B}\right)\\
&\qquad(\text{induction hypothesis})\\
=&\,[\![P]\!]\left(\left\langle \left(0,q^{i}X^{-i}F^{0}\right)+\left(0,(1-q)\cdot q^{i-1}X^{-i}F^{1}\right)\right\rangle_{B}\right)\\
&\qquad(B=(F=0))\\
=&\,[\![P]\!]\left(\left(0,q^{i}X^{-i}F^{0}\right)\right)\\
&\qquad(P=(X:=X-1;\{\mathit{skip}\}[q]\{F:=1\}))\\
=&\left(0,q^{i+1}X^{-(i-1)}F^{0}\right)+\left(0,(1-q)\cdot q^{i}X^{-(i+1)}F^{1}\right)
\end{aligned}$$

The intermediate claim is proven by induction. The final result is now obtained by restriction to $\neg B$.

$$\begin{aligned}
&\left\langle \mathrm{wh}_{B,P}^{i}(1)\right\rangle_{\neg B}\\
=&\left\langle \left(0,q^{i}X^{-i}F^{0}\right)+\left(0,(1-q)\cdot q^{i-1}X^{-i}F^{1}\right)\right\rangle_{\neg B}\\
=&\left(0,(1-q)\cdot q^{i-1}X^{-i}F^{1}\right)
\end{aligned}$$

□

8.9 Weak and Strong Bisimulations

Lemma 8.7. *Let B be a boolean condition, P a program and R a basic bisimulation. Then*

$$(K,G)\in R \implies \left(K-\sum_{i=1}^{n}\left\langle \mathrm{wh}_{B,P}^{i}(G)\right\rangle_{\neg B},\mathrm{wh}_{B,P}^{n}(G)\right)\in R \text{ for all } n\in\mathbb{N}$$

Proof. Proof by induction over n.
Base case for $n=0$:

$$\begin{aligned}
&(K,G)\in R\\
\implies&\,(K-0,G)\\
&\qquad(\text{empty sum})\\
\implies&\left(K-\sum_{i=1}^{0}\left\langle \mathrm{wh}_{B,P}(G)\right\rangle_{\neg B},\mathrm{wh}_{B,P}^{0}(G)\right)
\end{aligned}$$

Induction hypothesis:

$$(K, G) \in R \implies \left(K - \sum_{i=1}^{n} \left\langle \mathrm{wh}^{i}_{B,P}(G) \right\rangle_{\neg B}, \mathrm{wh}^{n}_{B,P}(G) \right) \in R \text{ for some } n \in \mathbb{N}$$

Inductive step:
R is a basic bisimulation, so we can apply the iteration step to the induction hypothesis which yields

$$\begin{aligned}
& (K, G) \in R \\
& \qquad \text{(induction hypothesis)} \\
\implies & \left(K - \sum_{i=1}^{n} \left\langle \mathrm{wh}^{i}_{B,P}(G) \right\rangle_{\neg B}, \mathrm{wh}^{n}_{B,P}(G) \right) \in R \\
& \qquad \text{(property ii) of weak bisimulations)} \\
\implies & \left(K - \sum_{i=1}^{n} \left\langle \mathrm{wh}^{i}_{B,P}(G) \right\rangle_{\neg B} - \left\langle \mathrm{wh}_{B,P}(\mathrm{wh}^{n}_{B,P}(G)) \right\rangle_{\neg B}, \mathrm{wh}_{B,P}(\mathrm{wh}^{n}_{B,P}(G)) \right) \in R \\
\implies & \left(K - \sum_{i=1}^{n} \left\langle \mathrm{wh}^{i}_{B,P}(G) \right\rangle_{\neg B} - \left\langle \mathrm{wh}^{n+1}_{B,P}(G) \right\rangle_{\neg B}, \mathrm{wh}^{n+1}_{B,P}(G) \right) \in R \\
\implies & \left(K - \sum_{i=1}^{n+1} \left\langle \mathrm{wh}^{i}_{B,P}(G) \right\rangle_{\neg B}, \mathrm{wh}^{n+1}_{B,P}(G) \right) \in R
\end{aligned}$$

□

Theorem (continuing from p. **??**)**.** *Let B be a Boolean condition, P a program and R a weak bisimulation.*
Then

$$(K, G) \in R \wedge G \models B \implies [\![\mathit{while}\,(B)\{P\}]\!]\,(G) \sqsubseteq_D K$$

Proof.

$$
\begin{aligned}
& (K,G) \in R \wedge G \models B \\
& \qquad \text{(Lemma \textbf{??})} \\
\implies & \left(K - \sum_{i=1}^{n} \left\langle \mathrm{wh}^{i}_{B,P}(G) \right\rangle_{\neg B}, \mathrm{wh}^{n}_{B,P}(G) \right) \in R \\
& \wedge \langle G \rangle_{\neg B} = 0 \\
& \qquad \text{(property i) of weak bisimulations)} \\
\implies & \left\langle \mathrm{wh}_{B,P}\left(\mathrm{wh}^{n}_{B,P}(G)\right) \right\rangle_{\neg B} \sqsubseteq_D K - \sum_{i=1}^{n} \left\langle \mathrm{wh}^{i}_{B,P}(G) \right\rangle_{\neg B} \\
& \wedge \left\langle \mathrm{wh}^{0}_{B,P}(G) \right\rangle_{\neg B} = 0 \\
\implies & \left\langle \mathrm{wh}^{0}_{B,P}(G) \right\rangle_{\neg B} + \sum_{i=1}^{n} \left\langle \mathrm{wh}^{i}_{B,P}(G) \right\rangle_{\neg B} + \left\langle \mathrm{wh}^{n+1}_{B,P}(G) \right\rangle_{\neg B} \sqsubseteq_D K \\
\implies & \sum_{i=0}^{n+1} \left\langle \mathrm{wh}^{i}_{B,P}(G) \right\rangle_{\neg B} \sqsubseteq_D K \\
\implies & \sup_{n \in \mathbb{N}} \sum_{i=0}^{n+1} \left\langle \mathrm{wh}^{i}_{B,P}(G) \right\rangle_{\neg B} \sqsubseteq_D \sup_{n \in \mathbb{N}} K \\
\implies & [\![\mathit{while}\,(B)\{P\}]\!]\,(G) \sqsubseteq_D K
\end{aligned}
$$

□

Lemma 8.8. *Let B be a Boolean condition, P a program and R a strong bisimulation for B and P with constant ε. Then,*

$$
(K,G) \in R \implies |K| - \left| \sum_{i=1}^{n} \left\langle \mathrm{wh}^{i}_{B,P}(G) \right\rangle_{\neg B} \right| \leq (1-\varepsilon)^n \cdot |K| \text{ for all } n \in \mathbb{N}
$$

Proof. Proof by induction over n.
Base case for $n = 0$:

$$
|K| - \left| \sum_{i=1}^{0} \left\langle \mathrm{wh}^{i}_{B,P}(G) \right\rangle_{\neg B} \right| = |K| - |0| \leq 1 \cdot |K| = (1-\varepsilon)^0 \cdot |K|
$$

Induction hypothesis:

$$
(K,G) \in R \implies |K| - \left| \sum_{i=1}^{n} \left\langle \mathrm{wh}^{i}_{B,P}(G) \right\rangle_{\neg B} \right| \leq (1-\varepsilon)^n \cdot |K| \text{ for some } n \in \mathbb{N}
$$

Inductive step:

$$
\begin{aligned}
&(K, G) \in R \\
&\qquad \text{(Lemma ??)} \\
\implies &\left(K - \sum_{i=1}^{n+1} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B}, \mathrm{wh}^n_{B,P}(G) \right) \in R \\
&\qquad \text{(Property iii) of strong bisimulations)} \\
\implies &\left| \left\langle \mathrm{wh}_{B,P}\left(\mathrm{wh}^n_{B,P}(G)\right) \right\rangle_{\neg B} \right| \geq \varepsilon \cdot \left| K - \sum_{i=1}^{n} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \right| \\
&\qquad \text{(negate, add to both sides)} \\
\implies &\left(|K| - \left| \sum_{i=1}^{n} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \right| \right) - \left| \left\langle \mathrm{wh}^{n+1}_{B,P}(G) \right\rangle_{\neg B} \right| \\
&\quad \leq \left(|K| - \left| \sum_{i=1}^{n} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \right| \right) - \varepsilon \cdot \left(|K| - \left| \sum_{i=1}^{n} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \right| \right) \\
&\qquad \text{(merge sums, factor out)} \\
\implies &|K| - \left| \sum_{i=1}^{n+1} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \right| \leq (1-\varepsilon) \cdot \left(|K| - \left| \sum_{i=1}^{n} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \right| \right) \\
&\qquad \text{(induction hypothesis)} \\
\implies &|K| - \left| \sum_{i=1}^{n+1} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \right| \leq (1-\varepsilon) \cdot (1-\varepsilon)^n \cdot |K| = (1-\varepsilon)^{n+1} \cdot |K|
\end{aligned}
$$

□

Lemma 8.9. *Let G and G' be PGFs of k variables with coefficients μ_s and μ'_s for $s \in \mathbb{Z}^k$, respectively. Then*

$$G \sqsubseteq_D G' \ \wedge \ |G| = |G'| \implies G = G'$$

Proof. Let G and G' be PGFs as above such that $G \sqsubseteq_D G'$ and $|G| = |G'|$. Now assume $G \neq G'$. Then there exists $s \in \mathbb{Z}^k$ with $\mu_s \neq \mu'_s$. Since $G \sqsubseteq_D G'$ it follows that $\mu_s < \mu'_s$. Adding all other coefficients on both sides does not break the inequation as every coefficient of G is less or equal to the respective coefficient of G'. Hence,

$$|G| = \sum_{s \in \mathbb{Z}^k} \mu_s < \sum_{s \in \mathbb{Z}^k} \mu'_s = |G'|$$

This is a contradiction to the precondition, so $G = G'$. □

Theorem (continuing from p. **??**)**.** *Let B be a Boolean condition, P a program and R a strong bisimulation for B and P with constant ε. Then*

$$(K, G) \in R \wedge G \models B \implies K = [\![\mathit{while}\,(B)\{P\}]\!]\,(G)$$

Proof.

$$(K, G) \in R \wedge G \models B$$

(Lemma **??**)

$$\implies |K| - \left| \sum_{i=1}^{n} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \right| \leq (1-\varepsilon)^n \cdot |K| \wedge \left\langle wh^0(G) \right\rangle_{\neg B} = 0$$

$$\implies |K| - \left| \sum_{i=0}^{n} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \right| \leq (1-\varepsilon)^n \cdot |K|$$

$$\implies \lim_{n\to\infty} \left(|K| - \left| \sum_{i=0}^{n} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \right| \right) \leq \lim_{n\to\infty} (1-\varepsilon)^n \cdot |K|$$

$$\implies \lim_{n\to\infty} |K| - \lim_{n\to\infty} \left| \sum_{i=0}^{n} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \right| \leq \lim_{n\to\infty} (1-\varepsilon)^n \cdot |K|$$

$a_n = \sum_{i=0}^{n} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B}$ is monotonic and bounded by its supremum given by the cpo D,

so a_n converges to its supremum.

$$\implies \lim_{n\to\infty} |K| - \left| \sup_{n\in\mathbb{N}} \sum_{i=0}^{n} \left\langle \mathrm{wh}^i_{B,P}(G) \right\rangle_{\neg B} \right| \leq \lim_{n\to\infty} (1-\varepsilon)^n \cdot |K|$$

$$\implies |K| - |[\![\mathit{while}\,(B)\{P\}]\!]\,(G)| \leq 0$$

$$\implies |K| \leq |[\![\mathit{while}\,(B)\{P\}]\!]\,(G)|$$

Using this and Theorem **??**, we get

$$(K, G) \in R \wedge G \models B$$

(apply Theorem **??** and the previous result)

$$\implies [\![\mathit{while}\,(B)\{P\}]\!]\,(G) \sqsubseteq_D K \wedge |K| \leq |[\![\mathit{while}\,(B)\{P\}]\!]\,(G)|$$

$$\implies |[\![\mathit{while}\,(B)\{P\}]\!]\,(G)| \leq |K| \wedge |K| \leq |[\![\mathit{while}\,(B)\{P\}]\!]\,(G)|$$

$$\implies |K| = |[\![\mathit{while}\,(B)\{P\}]\!]\,(G)|$$

To recap, from $(K, G) \in R \wedge G \models B$ it follows that

- $|[\![\mathit{while}\,(B)\{P\}]\!]\,(G)| = |K|$ from above, and

- $[\![\mathit{while}\,(B)\{P\}]\!]\,(G) \sqsubseteq_D K$ from Theorem **??**.

These satisfy the precondition of Lemma **??**, which gives us the final result

$$(K, G) \in R \wedge G \models B \implies K = [\![\mathit{while}\,(B)\{P\}]\!]\,(G) \qquad \square$$