From Logic Programs to Iterated Function Systems

—

Master's Thesis by Sebastian Bader

Supervisors: Prof. Dr. rer. nat. habil. Steffen Hölldobler
Dr. Pascal Hitzler

Department of Computer Science, Dresden University of Technology
Dresden, May 2003

Task

student: name: Sebastian Bader
date and place of birth: May 22nd 1977, Rostock
mat.-nr: 2509568

topic: Triggered by the desire to combine the advantages of logic programs and artificial neural networks, Hölldobler, Kalinke and Störr [HKS99] designed a mapping to embed interpretations into the space of real numbers. Using this mapping allows to obtain a real-valued graph of the $T_{\mathcal{P}}$-operator associated with the logic program. During the first study of these graphs it was found that they show self-similar structures like the ones created by iterated function systems.

The aim of this thesis is to investigate logic programs with respect to their transformability into iterated function systems.

goals:

- Review of the embedding of the $T_{\mathcal{P}}$-operator.
- Investigation of self-similar structures found in the graph.
- Modelling the structures using iterated function systems.
- Discussion of the consequences of the results obtained.

Prof. Dr. Steffen Hölldobler
(overseeing professor)

Dr. Pascal Hitzler
(supervisor)

Keywords

logic program, iterated function system, self-similar structure

Abstract

This thesis presents an approach which links logic programs and iterated function systems. The work was motivated by the discovery of self-similar structures during investigations of the $T_{\mathcal{P}}$-operator. This operator, associated with any program $\mathcal{P}$, was previously embedded into the real numbers. Therefore, it could be plotted as a graph in $\mathbb{R}^2$.

It is shown that for a large class of logic programs there exists an iterated function system such that the attractor coincides with the graph. An algorithm is given that constructs such systems for logic programs. For certain logic programs the correctness is proven. Furthermore, it is shown that for all acyclic programs with injective level mappings the graph can be approximated arbitrarily well.

In Section 6.1 possible applications of the theoretic results are presented. This includes an embedding into recurrent neural networks.

Zusammenfassung

In der vorliegenden Arbeit wird ein Ansatz dargestellt, der versucht, eine Verbindung zwischen logischen Programmen und Systemen iterierter Funktionen herzustellen. Motiviert wurde diese Arbeit durch die Entdeckung selbstähnlicher Strukturen während einer Untersuchung des $T_{\mathcal{P}}$-Operators. Dieser wurde zuvor in die reellen Zahlen eingebettet und konnte dadurch als Graph in $\mathbb{R}^2$ dargestellt werden.

Es wird gezeigt, dass für eine sehr große Klasse von Programmen Systeme iterierter Funktionen existieren, so dass der Graph des $T_{\mathcal{P}}$-Operators mit dem Attraktor übereinstimmt. Es wird ein Algorithmus angegeben, der für bestimmte logische Programme entsprechende Systeme konstruiert. Von diesem Algorithmus wird die Korrektheit bewiesen. Weiterhin wird gezeigt, dass für alle azyklischen Programme mit injektiver Stufenabbildung der Graph des $T_{\mathcal{P}}$-Operators beliebig genau approximiert werden kann.

In Kapitel 6.1 werden mögliche Anwendungen der gefundenen Resultate aufgezeigt. Diese schließen die Einbettung in rückgekoppelte neuronale Netze ein.

Contents

1. Introduction **1**
1.1. Initial Motivation . . . 1
1.2. Structure of the Thesis . . . 2
1.3. Looking into the Literature . . . 3

2. Logic Programs **4**
2.1. Syntax . . . 4
2.2. Semantics . . . 7

3. Iterated Function Systems **11**
3.1. Metric Spaces . . . 12
3.2. Contraction Mappings . . . 15
3.3. The Formal Definition . . . 17
3.4. Computing the Attractor . . . 21
3.5. Fractal Interpolation . . . 22

4. Graphs of Logic Programs **25**
4.1. Mapping Interpretations to Real Numbers . . . 25
4.2. Mapping Programs to Graphs in the Real Numbers . . . 31
4.3. A Generalised Fitting Metric . . . 32

5. From Logic Programs to Iterated Function Systems **35**
5.1. The General Structure and the Equivalence Theorem . . . 35
5.2. The Linear Case . . . 38
5.3. A Nonlinear Example . . . 46
5.4. Fractal Interpolation for Arbitrary Programs . . . 48

6. Applications, Results & Further Work **55**
6.1. Applications . . . 55
6.2. Results . . . 61
6.3. Further Work . . . 62

A. Attractors of Famous Iterated Function Systems **63**

B. Examples of Transformed Logic Programs **66**

C. Software **69**

D. Index of Definitions, List of Figures & References **72**

1. Introduction

Logic programs and artificial neural networks are important paradigms in computer science and in particular in artificial intelligence. But they are quite diverse. Logic programs are highly recursive and well understood from the declarative point of view. Neural networks, on the other hand, are highly parallel and adaptable but cannot be read declaratively, since the information is hidden somewhere in their connections. A system that is adaptive, robust, human readable and equipped with a declarative semantics, would be desirable. Such a system could be used to learn an unknown function with its neural network part and would provide a declarative reading of the underlying function.

To link both ideas Hölldobler et al developed a method to link the space of interpretations and the real numbers [HKS99]. This mapping was used to embed the $T_{\mathcal{P}}$-operator associated with a logic program into the reals. It was shown that this operator can be approximated using a feed-forward neural network. Unfortunately, there is no way known yet to construct such an approximating network directly from the program.

1.1. Initial Motivation

As mentioned above, Hölldobler et al developed a method to link the space of interpretations and the real numbers. This method was used to embed the immediate consequence operator $T_{\mathcal{P}}$ into the reals. Therefore, the graph of these real valued functions can be plotted as shown in Figure 1.

```
n(0).
n(s(X)) :- n(X).
```

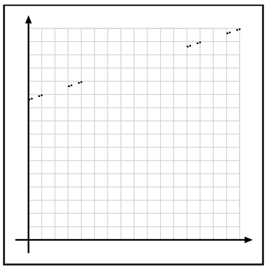


Figure 1: The graph of a real valued $T_{\mathcal{P}}$-operator

During the investigation of these plots, we found that they show self-similar properties, depicted as a zooming sequence in Figure 2. These structures are known as attractors of iterated function systems from fractal geometry. Motivated by this discovery, we studied

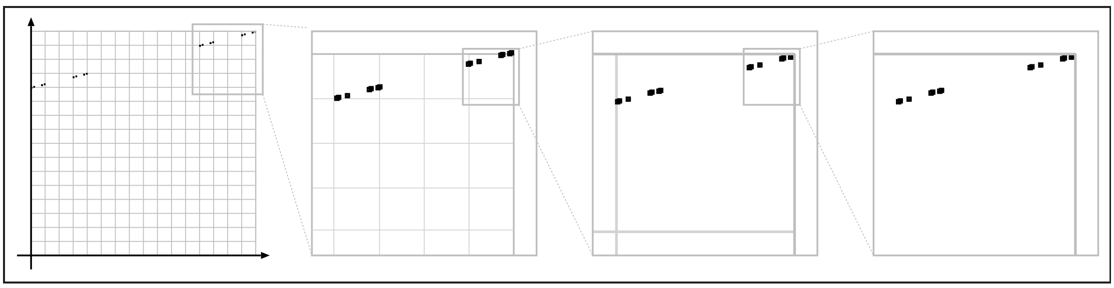


Figure 2: Self-similar structure of the graph

the relationship between logic programs and iterated function systems. It turned out that there is a deep, yet unknown relation between both systems. Therefore, we developed the necessary mathematics in order to create a unifying theory which helps to learn more about both systems and to understand their relation.

1.2. Structure of the Thesis

This thesis is organised as follows. First, logic programs and iterated function systems are introduced and most important notions recalled. Section 3 covers the necessary mathematics of e.g. complete metric spaces and contractive mappings.

Section 4 presents the approach by Hölldobler et al, which bridges the gap between the space of interpretations and the real numbers. A metric developed by Fitting is generalised and linked to the natural metric on the real numbers. Afterwards, the $T_{\mathcal{P}}$-operator associated with a logic program is embedded into the real numbers, its graph is plotted and properties of the graph are discussed.

In Section 5, ideas are presented which link logic programs and iterated function systems. As a main result, necessary and sufficient conditions are given under which the graph of a logic program coincides with the attractor of an iterated function system. A method is presented which constructs such a sytem and the correctness of the algorithm is shown for a certain subclass of programs. Afterwards the idea of fractal interpolation is applied to link both paradigms in more generality.

With a discussion of possible applications and further work the thesis finishes. Within the appendix examples of iterated function systems and transformated logic programs are given, as well as a description of the program used to create the plots.

This thesis is basically self contained, but background knowledge on logic programs and metric spaces would be helpful. Some proofs known from the literature, but of importance for this thesis, are only sketched and references are given.

Throughout this thesis the following notion is used:

`typewriter`	for source code of programs
sans serif	for indices, e.g. d_{f} is used for the metric on the space of functions
x, y	are used as super- and sub-scripts to indicate the x- and y-coordinate of the real plane, e.g. π_{x} denotes the projection to the x-coordinate

1.3. Looking into the Literature

The papers and books listed here were found useful by the author. The list is grouped in three parts, namely logic programming semantics, iterated function systems and the connection between logic programs and neural networks respectively. It is not meant to be a complete overview of the literature available, but to provide starting points for further reading.

Logic Programming and Negation: A Survey [AB94] This paper is a nice introduction into logic programming semantics. Different semantics are presented, discussed and related to one another. Furthermore, a comprehensive list of references is given.

Fractals Everywhere [Bar93] This book is a very detailed introduction into the theory of iterated function systems, including their mathematical background.

Undecidable Problems in Fractal Geometry [Dub95] Some questions concerning iterated function systems are shown to be undecidable. For example, the problem whether the attractor of a given system intersects the straight line from $(0,0)$ to $(1,1)$.

Metric Methods Three Examples and a Theorem [Fit94] Usually lattice-theoric arguments are used to show the existence of models for a given logic program. A new approach, which applies metric methods, is presented in this paper. The author defines a metric on the space of interpretation and shows that the $T_{\mathcal{P}}$-operator associated with an acyclic logic program is a contraction on this space. Therefore, using the Contraction Mapping Theorem by Banach, these programs possess a unique fixed point.

Towards a New Massively Parallel Computational Model for Logic Programming [HK94] By constructing a network computing the $T_{\mathcal{P}}$-operator for a given program and vice versa, the authors showed that a strong connection between propositional logic programs and recurrent neural networks of binary threshold units exits.

Approximating the Semantics of Logic Programs by Recurrent Neural Networks [HKS99] Using the embedding R (presented in Section 4.1) the authors bridge the gap between interpretations and real numbers. Furthermore, they could show that for acyclic logic programs the embedded $T_{\mathcal{P}}$-operator is a contraction and, therefore, continuous. They apply a theorem by Funahashi, that states that every continuous function on compact subsets can be approximated using feed-forward neural networks [Fun89], to show that the embedded $T_{\mathcal{P}}$-operator can be approximated arbitrarily well.

Continuity of Semantic Operators in Logic Programming and their Approximation by Artificial Neural Networks [HS01] The authors show that under certain conditions consequence operators are continuous with respect to the atomic topology $\mathcal{Q}$ on the space of interpretations. Using more general definitions of consequence operators and allowing multi-valued logics, the authors generalise results achieved in [HKS99]. Furthermore, they motivate the use of topological methods to control semantic operators.

2. Logic Programs

Originally developed for natural language processing, logic programs became an important paradigm in computer science and especially in artificial intelligence. They can be understood as a description of the world consisting of rules and facts.

Example 2.1 The natural numbers as defined by Peano written as a logic program.

```
nat_num(0).                        % 0 is a natural number

nat_num(s(X)) :- nat_num(X).       % the successor s(X) of a natural
                                   % number X is a natural number
```

This section does not give a detailed introduction to the theory of logic programs, but the most important notions are recalled to make this thesis self contained. First, the syntax of logic programs is introduced, i.e. the way a logic program is constructed. Afterwards, different semantics are briefly discussed.

2.1. Syntax

Logic programs are sets of formulae of first order logic. The alphabet of the language contains constants $c \in \mathcal{C}$, function symbols $f/n \in \mathcal{F}$ and predicate symbols $p/n \in \mathcal{R}$, where n denotes the respective arities. Furthermore, there are the usual connectives $\mathcal{J} = \{\neg/1, \wedge/2, \rightarrow/2\}$, the quantifiers $\{\forall, \exists\}$ and a set of first order variables $\mathcal{V}$. All these sets together are called the alphabet on which the first order language $\mathcal{L}(\mathcal{C}, \mathcal{F}, \mathcal{R})$ is defined. Since the sets of connectives, quantors and variables are assumed to be identical for every language, it is sufficient to name the constants, functions and relation symbols in order to define a language completely. A detailed introduction into the area can be found in [AP96, Höl01, Llo87].

Starting from an alphabet we can construct terms and atoms of the language. In a first order language, terms are constructed from function symbols, constants and variables, whereas atoms are composed using terms and predicate symbols.

Definition 2.2 (term, atom, literal) *A* term *of a first order language* $\mathcal{L}(\mathcal{C}, \mathcal{F}, \mathcal{R})$ *is recursively defined as either a variable* $v \in \mathcal{V}$*, a constant* $c \in \mathcal{C}$ *or an expression of the form* $f(t_1, \ldots, t_n)$*, where* $f/n \in \mathcal{F}$ *is an* n*-ary function symbol and* $t_1, \ldots, t_n$ *denote terms. An* atom *is defined as an expression of the form* $p(t_1, \ldots, t_n)$*, where* $p/n \in \mathcal{R}$ *is a predicate symbol and* $t_1, \ldots, t_n$ *are terms. A* literal *is either an atom or a negated atom, which is denoted* A *and* $\neg A$ *respectively.*

As mentioned above, logic programs consist of rules and facts. A rule contains a set of preconditions B and a conclusion A. Statements like "if it is hot and sunny, I will go swimming" are valid rules. This type of formula is usually called *Horn clause* or, in the context of logic programs, simply clause. Sometimes, A is referred to as *head* of the clause and B as the *body*.

Definition 2.3 (clause, fact) *Let A be an atom and B be a finite set of literals $\{L_1, L_2, \ldots, L_n\}$. Then a formula of the form $A \leftarrow L_1 \wedge L_2 \wedge \ldots \wedge L_n$ is called a* clause*. If B is the empty set, then the clause is called a* fact*, which is denoted $(A \leftarrow)$.*

Definition 2.4 (definite clause) *Let $A \leftarrow L_1 \wedge L_2 \wedge \ldots \wedge L_n$ be a clause. Then the clause is called* definite *if there is no negated literal in $\{L_1 \ldots L_n\}$.*

Within this thesis Prolog syntax is used, i.e. rules are written as $\mathtt{a} : - \mathtt{l1}, \mathtt{l2}, \ldots, \mathtt{lN}$ and facts as `a`.

Definition 2.5 (program) *Let $\mathcal{P} = \{C_1, \ldots, C_n\}$ be a finite set of clauses as defined above. Then $\mathcal{P}$ is called a* program*.*

Definition 2.6 (definite program, normal program) *Let $\mathcal{P}$ be a program. If all clauses in $\mathcal{P}$ are definite clauses, then $\mathcal{P}$ is called a* definite*, otherwise* normal*.*

As mentioned above, a program is defined on an alphabet $\mathcal{A}$. But each program implicitely defines an alphabet $\mathcal{A}$ and, therefore, the corresponding first order language.

Example 2.7 "The even and the odd numbers" as first order formula F and logic program $\mathcal{P}$. And the corresponding first order language $\mathcal{L}(\mathcal{C}, \mathcal{F}, \mathcal{R})$.

$$F = \forall x. \big([e(0)] \wedge [\neg e(x) \rightarrow e(s(x))] \wedge [e(x) \rightarrow o(s(x))]\big)$$

$\mathcal{P}$:

```
e(0).
e(s(X)) :- not e(X).
o(s(X)) :- e(X).
```

$\mathcal{L}(\mathcal{C}, \mathcal{F}, \mathcal{R})$:

$\mathcal{C} = \{0\}$
$\mathcal{F} = \{s/1\}$
$\mathcal{R} = \{e/1, o/1\}$

An important syntactic property of terms, clauses or programs is whether they contain variables or not. If they do not contain any variable, they are said to be *ground*.

Definition 2.8 (ground) *A term, atom, literal, clause or program is said to be* ground *if it does not contain any variable.*

The set of all ground terms $\mathcal{U}_\mathcal{P}$ contains all those terms that are constructed without any variable, i.e. using constants and function symbols only. The ground instances of a given atom, clause or program are computed by replacing every variable with every possible ground term.

Definition 2.9 (Herbrand base $\mathcal{B}_\mathcal{P}$, Herbrand universe $\mathcal{U}_\mathcal{P}$) *Let $\mathcal{P}$ be a logic program defined on the language $\mathcal{L}(\mathcal{C}, \mathcal{F}, \mathcal{R})$. Then the set of all ground terms is called the* Herbrand universe *$\mathcal{U}_\mathcal{P}$ and the set of all ground atoms is called the* Herbrand base *$\mathcal{B}_\mathcal{P}$.*

In Example 2.10 the Herbrand base and the Herbrand universe for the example above are shown, as well as parts of the grounded program.

Example 2.10 Given the program and language as in example 2.7.

Herbrand universe $\mathcal{U}_\mathcal{P}$:	$0, s(0), s(s(0)), \ldots$
Herbrand base $\mathcal{B}_\mathcal{P}$:	$e(0), o(0), e(s(0)), o(s(0)), e(s(s(0))), \ldots$
ground program *ground*($\mathcal{P}$):	`e(0).`
	`e(s(0)) :- not e(0).`
	`e(s(s(0))):- not e(s(0)).`
	$\vdots$
	`o(s(0)) :- e(0).`
	`o(s(s(0))):- e(s(0)).`
	$\vdots$

In the sequel of this thesis we will often need to enumerate the Herbrand universe or to identify its elements with natural numbers. This is done via so-called level mappings.

Definition 2.11 (level mapping) *Let $\mathcal{B}_\mathcal{P}$ be a Herbrand base. Then a function $|\cdot| : \mathcal{B}_\mathcal{P} \to \mathbb{N}$ is called a* level mapping *for $\mathcal{B}_\mathcal{P}$.*

Example 2.12 The "even and odd numbers" program and a level mapping:

$\mathcal{P}$:

```
e(0).
e(s(X)) :- not e(X).
o(s(X)) :- e(X).
```

$|\cdot|$:

$$|A| = \begin{cases} 2 \cdot n + 1 & \text{if } A = e(s^n(0)) \\ 2 \cdot n + 2 & \text{if } A = o(s^n(0)) \end{cases}$$

$$\begin{array}{rr} |e(0)| = 1 & |o(0)| = 2 \\ |e(s(0))| = 3 & |o(s(0))| = 4 \\ |e(s(s(0)))| = 5 & |o(s(s(0)))| = 6 \\ \vdots & \vdots \end{array}$$

Level mappings are often used for characterising logic programs or for defining subclasses of programs [ABW88, vGRS91, HW02, AP93], like for example the acyclic ones.

Definition 2.13 (acyclic logic program) *Let $\mathcal{P}$ be a logic program and $|\cdot|$ be a level mapping. If for all clauses $A \leftarrow L_1 \wedge L_2 \wedge \ldots \wedge L_n \in ground(\mathcal{P})$ and all $1 \leq i \leq n$ we have that $|A| > |L_i|$, then $\mathcal{P}$ is called acyclic with respect to $|\cdot|$. A program is called* acyclic*, if there exists such a level mapping.*

Since level mappings define an order on the atoms of the Herbrand base, they can easily be used to prove termination [AP93, Bez93] or the existence of models [Cav91, Fit02, LMPS95]. Furthermore, they were used by Fitting to apply metric methods within the area of logic programs [Fit94]. Hölldobler, Kalinke and Störr used level mappings to relate logic programs and connectionist models [HKS99]. Seda and Hitzler used them, for example, to apply topological methods [SH99, HW02, Hit01], which helped to generalise some of the results achieved before. Another recent application can be found in [HW02] where different semantics were characterised using level mappings.

2.2. Semantics

A semantics assigns a meaning to a logic program, i.e. how syntactic constructions relate to objects in the real world and their relations. There are two major types of semantics, namely the declarative and the operational one.

The declarative semantics gives a mathematically precise definition of the meaning of the program, e.g. the minimal set of all true sentences with respect to a given program $\mathcal{P}$. The operational or procedural semantics is usually defined as a mechanism with some user-input, i.e. it specifies the way answers are computed. Declarative semantics can be used to prove the correctness of such a mechanism.

First, the concepts of interpretations and models are introduced and, afterwards, the $T_{\mathcal{P}}$-operator is defined. Finally, some declarative semantics for logic programs will be briefly discussed.

2.2.1. Truth Values, Interpretations and Models

An interpretation is a mapping which assigns a truth value to each formula. In this thesis we will restrict ourselves to two-valued truth, i.e. we will talk about *true* and *false* only. Furthermore, we will restrict ourselves to Herbrand interpretations.

Unfortunately, even for a finite set of atoms, there are infinitely many formulae and to define such a function could be quite complicated. Fortunately, it is sufficient to assign truth values to atoms only, because the value for formulae is uniquely determined by means of the junctors. See e.g. [Höl01] for more detailed descriptions.

Definition 2.14 (Herbrand interpretations) *Let $\mathcal{B}_{\mathcal{P}}$ be the Herbrand base for a first-order language $\mathcal{L}(\mathcal{C}, \mathcal{F}, \mathcal{R})$. Then a subset $I \subseteq \mathcal{B}_{\mathcal{P}}$ is called a 2-valued* Herbrand interpretation*. It is assumed that to every atom $A \in I$ the truth value* true *is assigned and to every atom not contained in I the value* false*.*

Definition 2.15 (space of interpretations) *Let $\mathcal{B}_{\mathcal{P}}$ be a Herbrand base as defined in Definition 2.9. Then the set of all subsets of $\mathcal{B}_{\mathcal{P}}$, which is equivalent to the space of all interpretations, is denoted $I_{\mathcal{P}}$.*

If an interpretation assigns the value *true* to a formula, it is said to be a model for this formula. Since a logic program $\mathcal{P}$ is a conjunction of clauses, an interpretation is said to be a model for $\mathcal{P}$ if it maps every clause of $\mathcal{P}$ to *true*.

Definition 2.16 (Herbrand model) *Let I be a Herbrand interpretation and $\mathcal{P}$ be a logic program. If every clause in $ground(\mathcal{P})$ is evaluated to* true *under I, then I is said to be a* model *for P, which is denoted $I \models \mathcal{P}$.*

The declarative semantics of a program $\mathcal{P}$ is usually defined in terms of models of $\mathcal{P}$. In the sequel of this section different semantics are discussed and ways are shown to compute them, but first a major tool for the computation is introduced.

2.2.2. The Consequence Operator $T_{\mathcal{P}}$

This section introduces the $T_{\mathcal{P}}$-operator associated with every program $\mathcal{P}$. This operator plays a major role within this thesis. It maps a given interpretation I to the set of all its immediate consequences, with respect to $ground(\mathcal{P})$. Therefore, this operator is also called the *immediate consequence operator*.

Definition 2.17 ($T_{\mathcal{P}}$) *Let $\mathcal{P}$ be a logic program. Then the consequence operator $T_{\mathcal{P}}$ is defined as follows:*

$$\begin{aligned} T_{\mathcal{P}} : & I_{\mathcal{P}} \to I_{\mathcal{P}} \\ & I \mapsto \{A \mid \textit{there is a clause } (A \leftarrow Body) \textit{ in } ground(\mathcal{P}) \textit{ such that } I \models Body\} \end{aligned}$$

Example 2.18 Applications of the $T_{\mathcal{P}}$-operator for the "natural number" program:

$\mathcal{P}$:

```
n(0).
n(s(X)) :- n(X).
```

ground($\mathcal{P}$):

```
n(0).
n(s(0)) :- n(0).
n(s(s(0))) :- n(s(0)).
⋮
```

$$\begin{aligned} T_{\mathcal{P}}(\emptyset) &= \{n(0)\} \\ T_{\mathcal{P}}(\{n(0)\}) &= \{n(0), n(s(0))\} \\ T_{\mathcal{P}}(\{n(0), n(s(0))\}) &= \{n(0), n(s(0)), n(s(s(0)))\} \\ T_{\mathcal{P}}(\mathcal{B}_{\mathcal{P}}) &= \mathcal{B}_{\mathcal{P}} \end{aligned}$$

For every Herbrand interpretation I we have that $I \models \mathcal{P}$ if and only if $T_{\mathcal{P}}(I) \subseteq I$, i.e. all models are prefixed points of its associated $T_{\mathcal{P}}$-operator [AB94].

2.2.3. Different Semantics for Logic Programs

As mentioned above, the semantics of a logic program is defined in terms of its models. Within this section some semantics are briefly and informally discussed and their main motivations are given. Consider the following program $\mathcal{P}$:

```
eatable(X) :- fruit(X).            fruit(apple).
eatable(X) :- vegetable(X).        furniture(desk).
```

The underlying alphabet contains the set of constants $\mathcal{C} = \{\texttt{apple}, \texttt{desk}\}$, the predicate symbols $\mathcal{R}/n = \{\texttt{eatable}/1, \texttt{fruit}/1, \texttt{vegetable}/1, \texttt{furniture}/1\}$ and no function symbol. Possible models for this program are for example:

$$\begin{aligned} M_1 &= \{\texttt{fruit(apple)}, \texttt{furniture(desk)}, \texttt{eatable(apple)}\} \\ M_2 &= \{\texttt{fruit(apple)}, \texttt{furniture(desk)}, \texttt{eatable(apple)}, \texttt{fruit(desk)}, \texttt{eatable(desk)}\} \\ M_3 &= \mathcal{B}_{\mathcal{P}} \end{aligned}$$

Even if there is no evidence to include `fruit(desk)` and `eatable(desk)`, there is nothing to prevent it. Since M_1 is the "intended" model we need to distinguish M_1 from all other possible models, which led to the development of the minimal model semantics.

Least Model Semantics The idea is to minimise the positive information, i.e. we are interested in the minimal models, where a model is called minimal if there is no proper subset of it being a model itself. An atom is *true* according to the least model semantics if and only if it is contained in the least model. Emden and Kowalski showed in [vEK76] that for definite programs the least model exists and that it can be computed as the least fixed point of the $T_{\mathcal{P}}$-operator, which is reached starting with any interpretation and applying $T_{\mathcal{P}}$ repeatedly. For programs with negation this approach does not work, e.g. for the program `p :- not q.` there are two minimal models $\{\mathtt{p}\}$ and $\{\mathtt{q}\}$, therefore, no least model exists and the least model semantics is not applicable.

Supported model semantics This semantics is based on the idea that for every atom contained in a model there should be an explanation for it, i.e. I is called supported if and only if for all $A \in I$ there is a clause $A \leftarrow L_1 \wedge L_2 \wedge \ldots \wedge L_n$ in $ground(\mathcal{P})$ such that $I \models L_i$ for all L_i, hence $I = T_{\mathcal{P}}(I)$ [ABW88, AB94]. The supported model semantics does not define a way to compute the models, but for definite programs it obviously coincides with the least model semantics.

Clark's Completion Semantics The "Predicate Completion" was introduced by Clark in [Cla78] in order to define a semantics for normal programs. The underlying idea is to replace every *if* statement by an *if and only if*. Without going into the details the result of the procedure applied to the program above is:

$$\begin{aligned}
\mathtt{eatable}(X) &\leftrightarrow [\mathtt{fruit}(X) \vee \mathtt{vegetable}(X)] \\
\mathtt{fruit(X)} &\leftrightarrow [X = \mathtt{apple}] \\
\mathtt{furniture(X)} &\leftrightarrow [X = \mathtt{desk}]
\end{aligned}$$

The completion semantics of a program $\mathcal{P}$ is defined as the set models of $comp(\mathcal{P})$. Therefore, the semantics of `p :- not q.` is defined as $\{\{\mathtt{p}\}, \{\mathtt{q}\}\}$. Unfortunately, some consistent programs become inconsistent by completion. For example, $\mathtt{p} \leftarrow \mathtt{not\ p}$ becomes $\mathtt{p} \leftrightarrow \mathtt{not\ p}$ which is obviously inconsistent. In [AB94] it was shown that $I \models comp(\mathcal{P})$ if and only if $T_{\mathcal{P}}(I) = I$, i.e. the completion semantics of a program $\mathcal{P}$ coincides with the set of fixed points of the associated $T_{\mathcal{P}}$-operator. This semantics does not define a way to compute it either and coincides with the supported model semantics for all programs with consistent completion.

Fitting Semantics The inconsistency-problem of the completion semantics can be eliminated using three-valued models. Without going into the formal details, the three truth values *true*, *false* and *undefined* are used. Therefore, interpretations are redefined as a set of positive atoms together with a set of negative atoms. Furthermore, the immediate consequence operator $T_{\mathcal{P}}$ is adjusted to cope with this type of interpretations. The adjusted version, denoted $\Phi_{\mathcal{P}}$, possesses a least fixed point with respect to the information ordering [Prz89]. This fixed point is called the Fitting model of the program $\mathcal{P}$. The details can be found in [Fit85]. Again without going into details it turned out that neither the completion semantics nor the Fitting semantics can cope with "recursion through negation".

Stable Model Semantics The Gelfond-Lifschitz transformation of a program $\mathcal{P}$ with respect to an interpretation I is defined in two steps as follows:

1. Remove from $\mathcal{P}$ all clauses with $\neg A$ and $I \models A$.
2. Remove from the remaining clauses all negative literals.

Since the result $\mathcal{P}_I$ is definite, it possesses a least model J and the Gelfond-Lifschitz operator Γ is defined as $\Gamma(I) = J$. The fixed points of the Gelfond-Lifschitz operator Γ are always models of the program $\mathcal{P}$ and are called stable models [GL88]. An atom is true according to the stable model semantics and a program $\mathcal{P}$, if and only if it is contained in all stable models of $\mathcal{P}$. Unfortunately, we have to guess the stable models, i.e. there is no way to compute them, but enumerating every possible interpretation and to check whether it is a stable model or not.

Well-founded Semantics The stable model semantics presented above allows any number of stable models for a logic program. This can be interpreted as uncertainty. Using one three-valued model, instead of multiple two-valued ones, the well-founded semantics avoids this problem [vGRS91]. There are multiple characterisations of this semantics [vGRS91, BF91]. One of them is closely related to the definition of the stable models semantics above [PP90]. Using an extended version Γ^* of the Gelfond-Lifschitz operator the well founded model is defined as the unique least fixed point of Γ^*. This semantics is applicable to all logic programs and provides a unique three-valued model for each. In a recent paper by Hitzler and Wendt [HW02] a new characterisation of different semantics is given. Using level mappings the authors show that the well-founded semantics is a stratified version of the Fitting semantics.

3. Iterated Function Systems

This section introduces iterated function systems, abbreviated IFS. For motivation, the (well known) Cantor set is discussed. It is a good introduction into this field, since it is the prototype of a fractal created by an iterated function system. The Cantor set is usually defined by one of the following constructions (Algorithm 3.1 and 3.2).

Algorithm 3.1 (Construction of the Cantor set $\mathcal{C}$ I)

1. Take the interval $[0, 1]$ from the real line and call it C_1.
2. Remove from C_1 the middle third which gives C_2.
3. Construct C_{n+1} by removing the middle thirds from the remaining parts in C_n.
4. Continue with step 3.

This process is continued ad infinitum, where $\mathcal{C} = \cap C_i$ is called the Cantor set. The first 5 steps are shown in figure 3.

Algorithm 3.2 (Construction of the Cantor set $\mathcal{C}$ II)

1. Take the interval $[0, 1]$ from the real line and call it C_1.
2. Create two copies of C_n and shrink both to one third of their size.
3. Create a new image with the two copies placed on the left and the right side.
4. Continue with step 2.

Again this process is continued ad infinitum and $\mathcal{C} = \cap C_i$ is called the Cantor set.

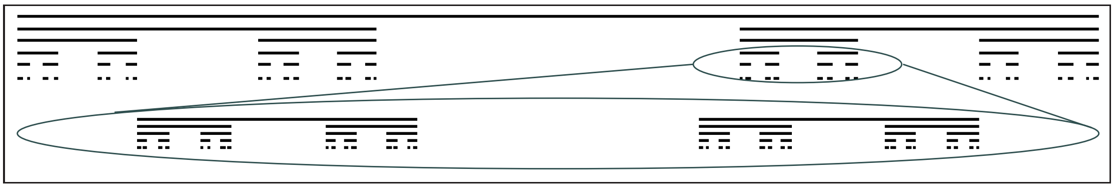


Figure 3: Five steps of the production of the Cantor set $\mathcal{C}$

Without knowing the exact definitions yet, Algorithm 3.2 describes an *Iterated Function System* (IFS) and a way to compute its so-called *Attractor* $\mathcal{C}$. Those systems consist of a set of contractive mappings defined on a complete metric space, which are repeatedly applied to a given "starting"-set. The "final" image of this process is called the attractor, which is a *fractal*. Fractals are (informally) defined as pictures with a feature called self-similarity. "Zooming into" the pictures leads to identical or similar structures again and again. Within this thesis the word fractal is used for a member of the set $\mathcal{H}(\mathbf{X})$, which is introduced in Section 3.3.

This section gives first a short introduction into metric spaces and contractive mappings. Afterwards, iterated function systems are introduced formally and, finally, the idea of fractal interpolation is presented. The whole section follows the notations of [Bar93], which is used as main reference without further notice.

3.1. Metric Spaces

In mathematics the term *space* is usually used to refer to a set with some additional structure, where structure means that there are some kind of relations between the members of the set. If the expressions "close to" and "far from" make sense for a given set, this set could be a metric space. A metric on a set is a function measuring the distance between members of the set and hence creating a structure on it. If some properties concerning this function hold, the set together with the function is said to be a metric space.

Definition 3.3 (metric space) *Let $\mathbf{X}$ be a set and let $d : \mathbf{X} \times \mathbf{X} \to \mathbb{R}$ be a function. Then $(\mathbf{X}, d)$ is called a* metric space *if the following conditions hold for all $x, y, z \in \mathbf{X}$:*

(i) $d(x, x) = 0$
(ii) $d(x, y) = d(y, x)$
(iii) $d(x, z) \leq d(x, y) + d(y, z)$ *(known as the triangular inequality)*

Then the function d is called a metric *on $\mathbf{X}$.*

Example 3.4 (metric spaces)

set	metric
real numbers ($\mathbb{R}$)	$d(x, y) = \lvert x - y \rvert$
Euclidean plane ($\mathbb{R}^2$)	$d\big((x_1, y_1)(x_2, y_2)\big) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$
space of interpretations ($I_{\mathcal{P}}$)	$d_{\mathcal{P}}(I, J) = 2^{-n}$ where I and J differ on an atom with level n, but agree on all atoms with level $< n$

The following three definitions are used to link different metric spaces in a certain way. This can be done, if there is a similar structure on both. Consider the two metric spaces $(\mathbf{X}, d_1)$ and $(\mathbf{X}, d_2)$. If all points which are close to each other using d_1 are close to each other using d_2, then these two spaces might be equivalent.

Definition 3.5 (equivalent metrics) *Two metrics d_1 and d_2 defined on the space $\mathbf{X}$ are said to be* equivalent *if there exist real constants $0 < c_1 < c_2 < \infty$ such that for all $x, y \in \mathbf{X}$ we have that $c_1 \cdot d_1(x, y) \leq d_2(x, y) \leq c_2 \cdot d_1(x, y)$ holds.*

Definition 3.6 (equivalent metric spaces) *Two metric spaces $(\mathbf{X}, d_{\mathbf{X}})$ and $(\mathbf{Y}, d_{\mathbf{Y}})$ are said to be* equivalent *if there is a bijective function $h : \mathbf{X} \to \mathbf{Y}$, such that the metric d on $\mathbf{X}$ defined by $d(x_1, x_2) = d_{\mathbf{Y}}(h(x_1), h(x_2))$ is equivalent to $d_{\mathbf{X}}$.*

Definition 3.7 (isometric spaces) *A metric space $(\mathbf{X}, d_{\mathbf{X}})$ is said to be isometric to some metric space $(\mathbf{Y}, d_{\mathbf{Y}})$, if there is a bijection $f : \mathbf{X} \to \mathbf{Y}$ which preserves the distances, i.e. $d_{\mathbf{X}}(x_1, x_2) = d_{\mathbf{Y}}(f(x_1), f(x_2))$, for all $x_1, x_2 \in \mathbf{X}$.*

As there are distances defined between the points of the set, it is possible to speak about convergence of sequences, i.e. whether the elements get closer to some point.

Definition 3.8 (convergence, limit point) *A sequence of points $x_1, x_2, x_3, \ldots$ in a metric space $(\mathbf{X}, d)$ converges to a point $x \in \mathbf{X}$, if for every $\epsilon > 0$ there is an element x_n of the sequence such that for all following elements x_i with $i > n$ the distance $d(x_i, x) < \epsilon$. The point x is called the* limit point *of the sequence. A convergent sequence is denoted $x_1, x_2, x_3, \ldots \rightarrow x$.*

Example 3.9 ((non)-converging sequences)

sequence	limit point
$\frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$	0
$1, 2, 3, 4, \ldots$	not converging
$\left(1+\frac{1}{1}\right)^1, \left(1+\frac{1}{2}\right)^2, \left(1+\frac{1}{3}\right)^3, \left(1+\frac{1}{4}\right)^4, \ldots$	$e = 2.718\ldots$

Definition 3.10 (Cauchy sequence) *Let $(\mathbf{X}, d)$ be a metric space. A sequence $x_1, x_2, \ldots$ is called a* Cauchy sequence *or* Cauchy*, if for any given number $\epsilon > 0$ there is an integer N such that for all $n, m > N$ we have that $d(x_n, x_m) < \epsilon$.*

The fact that a sequence is Cauchy in $(\mathbf{X}, d)$ does not ensure that it has a limit point in $\mathbf{X}$. Maybe the sequence is approaching a point, which does not belong to the set. For example the third sequence in Example 3.9 defined on the space of rational numbers converges to e, which is not rational itself. If all Cauchy sequences of a given metric space $\mathbf{X}$ have their limit point in $\mathbf{X}$, this space is called complete. Intuitively speaking a complete metric space does not have any holes, i.e. there are no points missing.

Definition 3.11 (complete) *Let $(\mathbf{X}, d)$ be a metric space. If every Cauchy sequence has a limit point in $\mathbf{X}$, then $(\mathbf{X}, d)$ is said to be* complete.

Definition 3.12 (bounded) *Let $(\mathbf{X}, d)$ be a metric space. A subset $S \subseteq \mathbf{X}$ is said to be* bounded*, if there is a real number D, such that for all $s_1, s_2 \in S$ we have $d(s_1, s_2) \leq D$.*

Definition 3.13 (closed) *A subset S of a metric space $(\mathbf{X}, d)$ is said to be* closed*, if every convergent sequence $s_1, s_2, \ldots$ with $s_i \in S$ has its limit point in S.*

Proposition 3.14 *Let $(\mathbf{X}, d)$ be a complete metric space and let $S \subseteq \mathbf{X}$ be closed. Then (S, d) is a complete metric space.*

> **Proof** Let $s_1, s_2, s_3, \ldots$ be a Cauchy sequence in S and, therefore, in $\mathbf{X}$. From the completeness of $\mathbf{X}$ we know that there is a $x \in \mathbf{X}$, which is the limit point of the sequence. As all elements of the sequence are contained in S, which is known to be closed, the limit x must be contained in S as well. This completes the proof. □

Definition 3.15 (compact) *Let $S \subseteq \mathbf{X}$ be a subset of a metric space $(\mathbf{X}, d)$. S is said to be* compact *if every sequence in S contains a subsequence with a limit in S.*

Looking at the definitions above one can relate compactness, completeness and closedness to each other as follows: compact $\Rightarrow$ complete $\Rightarrow$ closed. In other words every compact subset of a metric space is complete and every complete subset is closed.

The following definitions and propositions are concerned with properties of mappings. A very important one is the so-called continuity of a function. Informally it says, that small changes in x change the image $f(x)$ only a little.

Definition 3.16 (continuous) *Let f be a mapping from $(\mathbf{X}, d_{\mathbf{X}})$ to $(\mathbf{Y}, d_{\mathbf{Y}})$. Then f is said to be* continuous *at x, if for any $\epsilon > 0$ there is a $\delta > 0$, such that for all $y \in \mathbf{X}$ we have that $d_{\mathbf{X}}(x, y) < \delta$ implies that $d_{\mathbf{Y}}(f(x), f(y)) < \epsilon$. Alternatively, f is* continuous *at x if for any sequence $x_1, x_2, \ldots \to x$ the sequence $f(x_1), f(x_2), \ldots$ converges to $f(x)$.*

Definition 3.17 (continuous) *A mapping $f : \mathbf{X} \to \mathbf{Y}$ is said to be* continuous, *if and only if it is continuous at every point $x \in \mathbf{X}$.*

Definition 3.18 (uniformly continuous) *A mapping $f : \mathbf{X} \to \mathbf{Y}$ is said to be* uniformly continuous, *if for any $\epsilon > 0$ there is a $\delta > 0$, such that for all $x, y \in \mathbf{X}$ we have that $d_{\mathbf{X}}(x, y) < \delta$ implies that $d_{\mathbf{Y}}(f(x), f(y)) < \epsilon$.*

Proposition 3.19 *Let $(\mathbf{X}, d_{\mathbf{X}})$, $(\mathbf{Y}, d_{\mathbf{Y}})$ and $(\mathbf{Z}, d_{\mathbf{Z}})$ be complete metric spaces and let $f : \mathbf{X} \to \mathbf{Y}$ and $g : \mathbf{Y} \to \mathbf{Z}$ be continuous. Then $g \circ f$ is continuous.*

Proof As g is continuous, we know that for any $\epsilon > 0$ there is a $\beta > 0$ such that $d_Y(y_1, y_2) < \beta$ implies $d_{\mathbf{Z}}(g(y_1), g(y_2)) < \epsilon$. Furthermore, there is a $\delta > 0$ for every β such that $d_X(x_1, x_2) < \delta$ implies $d_{\mathbf{Y}}(f(x_1), f(x_2)) < \beta$, since f is continuous. Therefore, we can conclude that for any $\epsilon > 0$ there is a $\delta > 0$ such that such that $d_X(x, y) < \delta$ implies $d_Z(g(f(x_1)), g(f(x_2))) < \epsilon$, what shows that $g \circ f$ is continuous. □

Proposition 3.20 *Let $f : \mathbf{X} \to \mathbf{Y}$ be a continuous mapping and $\mathbf{X}$ be compact. Then $f(\mathbf{X})$ is compact.*

Proof For every sequence $x_1, x_2, \ldots \in \mathbf{X}$ there is a converging subsequence $x_{k_1}, x_{k_2}, \ldots$ with limit point $x \in \mathbf{X}$. From the continuity of f it follows, that the sequence $y_{k_i} = f(x_{k_i})$ converges to $y = f(x)$. Therefore, $f(\mathbf{X})$ must be compact. □

Corollary 3.21 *Let $f : \mathbf{X} \to \mathbf{Y}$ be continuous and onto and let $\mathbf{X}$ be compact. Then $\mathbf{Y}$ is compact.*

One of the most important metrics is the *Euclidean metric*, defined on the Euclidean space. This is the metric we use in our daily life, which computes the length of the direct connection between two points. It is often referred to as the *natural metric* on the real numbers.

Definition 3.22 (Euclidean Metric) *Let $\mathbb{R}^n$ be the n-dimensional real space with $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$. Then $d_n(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ is called the* Euclidean metric.

The following two propositions introduce two well known complete metric spaces, namely the n-dimensional real space and the space of continuous functions on compact sets. Both proofs can be found in standard books on analysis, e.g. [Kön02].

Proposition 3.23 *The n-dimensional Euclidean metric is a metric on $\mathbb{R}^n$ as defined in Definition 3.3. And $(\mathbb{R}^n, d_n)$ is a complete metric space.*

Proposition 3.24 *Let $\mathcal{F}$ denote the set of continuous functions from K to $\mathbb{R}$, where K is a compact subset of R and let d_f be defined as $d_f(f, g) = \max\{d_1(f(x), g(x)) \mid x \in K\}$, where d_1 denotes the 1-dimensional Euclidean metric. Then $(\mathcal{F}, d_f)$ is a complete metric space.*

3.2. Contraction Mappings

In the sequel of this thesis we will often talk about contractive mappings. The intuition behind such a mapping is that it “shrinks” some subset of $\mathbf{X}$. Contractive mappings posses some nice properties like converging to unique fixed points and being continuous. In Section 3.3 we will find that iterated function systems are contractive mappings on a certain space.

Definition 3.25 (contraction) *A mapping $f : \mathbf{X} \to \mathbf{X}$ on a metric space $(\mathbf{X}, d)$ is called* contractive *or a* contraction mapping*, if there is a real constant s such that $0 \leq s < 1$ and for all $x, y \in \mathbf{X}$ we have $d(f(x), f(y)) \leq s \cdot d(x, y)$. Any such number s is called a* contractivity factor *for f.*

Example 3.26 (contractive mappings) The following mappings are contractions.

metric space	contraction
real numbers $(\mathbb{R}, d_1)$	$f(x) = \frac{1}{2}x + 3$
Euclidean plane $(\mathbb{R}^2, d_2)$	$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0.5 \\ 2 \end{pmatrix}$
interpretations $(I_{\mathcal{P}}, d_{\mathcal{P}})$	$f(I) = \{A \mid \text{there is a } B \in I \text{ with } \lvert A\rvert = \lvert B\rvert + 2\}$

Proposition 3.27 *Let $(\mathbf{X}, d)$ be a metric space and let $f : \mathbf{X} \to \mathbf{X}$ be a contractive mapping with contractivity factor s. Then f is continuous on $(\mathbf{X}, d)$.*

Proof Let $\delta = \epsilon/s$. Since $d(f(x), f(y)) \leq s \cdot d(x, y) < \epsilon$ holds for all $x, y \in \mathbf{X}$ with $d(x, y) < \delta$, f is a contraction. □

Contractive mappings on complete metric spaces are a powerful tool and are used to solve various problems. As stated by Banach, for every contractive mapping f the sequence $f(x), f(f(x)), f(f(f(x))), \ldots$ converges to the same unique fixed point. Therefore, contractions can be used to find fixed points or to prove their existence.

Theorem 3.28 (Banach Contraction Mapping Principle) *Let $f : \mathbf{X} \to \mathbf{X}$ be a contractive mapping on the complete metric space $(\mathbf{X}, d)$. Then f possesses exactly one fixed point $x \in \mathbf{X}$ and for any point $y \in \mathbf{X}$ the sequence $\{f^n(y) \mid n = 0, 1, \ldots\}$ converges to x.*

Proof (sketch) The proof of this theorem can be done in three steps. First, it is shown that $f(x), f(f(x)), f(f(f(x))), \ldots$ is a Cauchy sequence. As $(\mathbf{X}, d)$ is complete this sequence converges, say to x_f. Afterwards, it is shown, that x_f is a fixed point, which follows from the continuity of f. Finally, it is left to show, that there is only one such x_f, which can be shown assuming that there are two distinct ones and contradicting this assumption. A complete proof of this theorem can be found in any book on analysis, e.g. [Kön02]. □

The Banach Contraction Mapping Principle is also known as the Banach Fixed Point Theorem. The theorem is used to prove the existence of attractors of iterated function systems and of uniquely determined fractal interpolation functions in Section 3.5.

The sequel of this section gives two propositions which relate different contractions. They use the idea of equivalent metric spaces to give conditions under which a contraction on one space is a contraction on another one as well. These propositions are used in Section 4.3 to prove that the embedded $T_{\mathcal{P}}$-operator of acyclic logic programs is a contraction on $(\mathbb{R}, d_1)$.

Proposition 3.29 (equivalent contractions I) *Let d_1 and d_2 be equivalent metrics on $\mathbf{X}$, connected via c_1 and c_2 as introduced in Definition 3.5. Let f be a contraction on $(\mathbf{X}, d_1)$ with contractivity factor s_f. Then f is a contraction on $(\mathbf{X}, d_2)$ with contractivity factor $\tilde{s}_f = \frac{c_2 \cdot s_f}{c_1}$, if $0 \leq \frac{c_2 \cdot s_f}{c_1} < 1$.*

Proof From the fact that d_1 and d_2 are equivalent it follows that $d_2(f(x), f(y)) \leq c_2 \cdot d_1(f(x), f(y))$. But as f is a contraction on $(\mathbf{X}, d_1)$, we know that $d_2(f(x), f(y)) \leq c_2 \cdot s_f \cdot d_1(x, y)$. Again by using the equivalence of d_1 and d_2 we can conclude that $d_2(f(x), f(y)) \leq \frac{c_2 \cdot s_f}{c_1} \cdot d_2(x, y)$, what ends the proof, because $0 \leq \frac{c_2 \cdot s_f}{c_1} < 1$ holds by assumption. Therefore, f is a contraction on $(\mathbf{X}, d_2)$ with contractivity factor $\tilde{s}_f = \frac{c_2 \cdot s_f}{c_1}$. □

Proposition 3.30 (equivalent contractions II) *Let $(\mathbf{X}_1, d_1)$ and $(\mathbf{X}_2, d_2)$ be equivalent metric spaces, connected via the mapping $h : \mathbf{X}_1 \to \mathbf{X}_2$ and the two constants c_1 and c_2 as introduced in Definition 3.6. Let f be a contraction on $(\mathbf{X}_1, d_1)$ with contractivity factor s_f. Let g be defined as $g : \mathbf{X}_2 \to \mathbf{X}_2 : x \mapsto h(f(h^{-1}(x)))$. Then g is a contraction on $(\mathbf{X}_2, d_2)$ with contractivity factor $s_g = \frac{c_2 \cdot s_f}{c_1}$, if $0 \leq \frac{c_2 \cdot s_f}{c_1} < 1$.*

Proof Using the definition of $g(x) = h(f(h^{-1}(x)))$ we get that $d_2(g(x), g(y)) = d_2(h(f(h^{-1}(x))), h(f(h^{-1}(y))))$. As introduced in Definition 3.6 $\tilde{d}_1$ is defined as $\tilde{d}_1(x, y) = d_2(h(x), h(y))$. Taking this into account, we get that $d_2(g(x), g(y)) = \tilde{d}_1(f(h^{-1}(x)), f(h^{-1}(y)))$. Knowing that f is a contraction on $(\mathbf{X}_1, d_1)$ and applying the result of the proposition above we derive $d_2(g(x), g(y)) \leq s_g \cdot \tilde{d}_1(h^{-1}(x), h^{-1}(y))$, where $s_g = \frac{c_2 \cdot s_f}{c_1}$. Finally, applying the definition of $\tilde{d}_1$, we have $d_2(g(x), g(y)) \leq s_g \cdot d_2(x, y)$. □

The following section will use the notations presented so far to define iterated function systems formally.

3.3. The Formal Definition

Definition 3.31 (IFS) *A (hyperbolic)* iterated function system (IFS) *consists of a complete metric space* $(\mathbf{X}, d)$ *together with a finite set* $\Omega = \{\omega_1, \dots, \omega_n\}$ *of contraction mappings* $\omega_i : \mathbf{X} \to \mathbf{X}$*, with respective contractivity factors* s_i*, for* $i = 1, 2, \dots, n$*. The contractivity factor of the IFS is defined as* $s = \max(\{s_i \mid 0 < i \leq n\})$*.*

The term *hyperbolic* in the definition above emphasises that all ω_i are contractions. As mentioned before the "final" image of the repetitive application of the mappings is a fractal, i.e. something with self-similar properties. Figure 4 shows the production of such a structure, the Sierpinski triangle. It is produced by an IFS consisting of the following three mappings on $\mathbb{R}^2$.

$$\omega_1\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} .5 & 0 \\ 0 & .5 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \omega_2\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} .5 & 0 \\ 0 & .5 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} .5 \\ 0 \end{pmatrix} \quad \omega_2\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} .5 & 0 \\ 0 & .5 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ .5 \end{pmatrix}$$

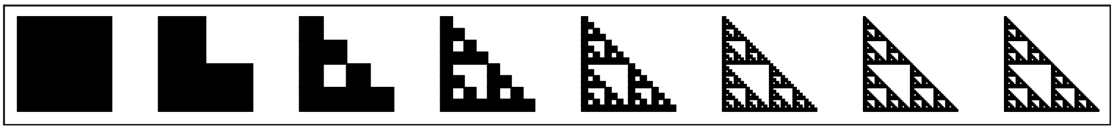


Figure 4: The first steps of the production of the Sierpinski triangle

In Figure 4 it seems to be obvious, that the process will lead to some kind of stable state or fixed point. To formalise this we will first introduce the space $\mathcal{H}(\mathbf{X})$. That is the space where the fractals "live". Furthermore, an appropriate metric h_d will be defined, which turns $(\mathcal{H}(\mathbf{X}), h_d)$ into a complete metric space. Afterwards, contractive mappings in $(\mathbf{X}, d)$ are extended to contractive mappings in $(\mathcal{H}(\mathbf{X}), h_d)$, which allows to apply Banach's theorem. Finally, a theorem by Barnsley is given, which states that there exists a fixed point A for every IFS, such that $\Omega(A) = A$. All proofs of this section are only sketched and can be found in [Bar93]. The sketches are included to make this thesis self contained and to convey the underlying ideas.

3.3.1. The Space of Fractals

Definition 3.32 ($\mathcal{H}(\mathbf{X})$) *Let* $(\mathbf{X}, d)$ *be a complete metric space. Then* $\mathcal{H}(\mathbf{X})$ *denotes the set of compact subsets of* $\mathbf{X}$*, other than the empty set.*

Definition 3.33 (Hausdorff distance) *Let* $(\mathbf{X}, d)$ *be a complete metric space,* $x \in \mathbf{X}$ *and* $A, B \in \mathcal{H}(\mathbf{X})$*. Then* $d(x, B) = \min\{d(x, y) \mid y \in B\}$ *is called the distance between the point* x *and the set* B*. The distance between the two sets* A *and* B *is defined as* $d(A, B) = \max\{d(a, B) \mid a \in A\}$*. The* Hausdorff distance h_d *between two sets* $A, B \in \mathcal{H}(\mathbf{X})$ *is defined as* $h_d(A, B) = \max(d(A, B), d(B, A))$*.*

For the sequel we assume a continuous mapping $f : \mathbf{X} \to \mathbf{X}$ to be extended to be functions on $\mathcal{H}(\mathbf{X})$ the usual way, i.e. $f(A) = \{f(a) \mid a \in A\}$.

Proposition 3.34 *Let* $(\mathbf{X}, d)$ *be a complete metric space,* $\mathcal{H}(\mathbf{X})$ *be defined as above and let* h_d *denote the Hausdorff distance constructed on d. Then* $(\mathcal{H}(\mathbf{X}), h_d)$ *is a complete metric space.*

> **Proof (sketch)** To show this we need to show that h_d is a metric on $\mathcal{H}(\mathbf{X})$, what can be done by showing that the three conditions stated in Definition 3.3 hold. As usual the first two follow immediately from the definition. The third can be shown by pushing the inequality down to the point-set-distance, where it can be validated easily. The existence of the minima and maxima is shown using the compactness and non-emptiness of the members of $\mathcal{H}(\mathbf{X})$.
> To prove the completeness it is necessary to show that there is a limit point in $\mathcal{H}(\mathbf{X})$ for every Cauchy sequence. Let A be the limit of the sequence $A_1, A_2, \ldots \in \mathcal{H}(\mathbf{X})$. It can be shown that for any $x \in A$ there is a Cauchy sequence $x_n \in A_n$ which converges to x. Furthermore, it is necessary to show that $A \in \mathcal{H}(\mathbf{X})$, which can be done by proving that A is bounded, closed and not empty. The full proof can be found in [Bar93, p. 27–37]. □

Proposition 3.35 *Let* $f : \mathbf{X} \to \mathbf{X}$ *be a contractive mapping on the metric space* $(\mathbf{X}, d)$ *with contractivity factor s. Then* $f : \mathcal{H}(\mathbf{X}) \to \mathcal{H}(\mathbf{X})$ *is a contractive mapping on* $(\mathcal{H}(\mathbf{X}), h_d)$ *with contractivity factor s.*

> **Proof (sketch)** This can be proven by showing that f maps $\mathcal{H}(\mathbf{X})$ into itself, which follows from the continuity of f and the compactness of the elements of $\mathcal{H}(\mathbf{X})$. Afterwards, it follows from the definition of h_d that f is a contraction on $(\mathcal{H}(\mathbf{X}), h_d)$. □

From the Banach Contraction Mapping Principle we know that a contractive mapping on a metric space possesses a unique fixed point. This can be extended to sets of contractive mappings in the space of fractals as stated in the following theorem.

Theorem 3.36 *Let* $\{(\mathbf{X}, d), \Omega\}$ *be a hyperbolic iterated function system with contractivity factor s. Its unique fixed point,* $A \in \mathcal{H}(\mathbf{X})$*, obeys* $A = \Omega(A)$ *and can be reached from all* $B \in \mathcal{H}(\mathbf{X})$ *as* $A = \lim_{n\to\infty} \Omega^n(B)$*, where the limit is taken with respect to* h_d*.*

> **Proof (sketch)** First, it is shown that the transformation $\Omega : \mathcal{H}(\mathbf{X}) \to \mathcal{H}(\mathbf{X})$ defined by $\Omega(B) = \cup_{i=1}^{n} \omega_i(B)$ for all $B \in \mathcal{H}(\mathbf{X})$, is a contraction mapping on the complete metric space $(\mathcal{H}(\mathbf{X}), h_d)$ with contractivity factor s. This includes showing that Ω maps $\mathcal{H}(\mathbf{X})$ into itself. Afterwards, the Banach Contraction Mapping Principle ensures that there is such a uniquely determined fixed point. □

Definition 3.37 (attractor) *The fixed point* $A \in \mathcal{H}(\mathbf{X})$ *described in Theorem 3.36 is called the* attractor *of the IFS* $\{(\mathbf{X}, d), \Omega\}$*, which is denoted by* $A(\{(\mathbf{X}, d), \Omega\})$*.*

Note that the attractor is a compact set, for it is an element of $\mathcal{H}(\mathbf{X})$. A more detailed explanation and the complete proof of Theorem 3.36 can be found in [Bar93, p. 79–81]. The theorem above states that the unique fixed point of a hyperbolic IFS exists and that it can be computed by repetitive application of the contractive mappings to an arbitrary “starting set”. This is depicted in figure 5.

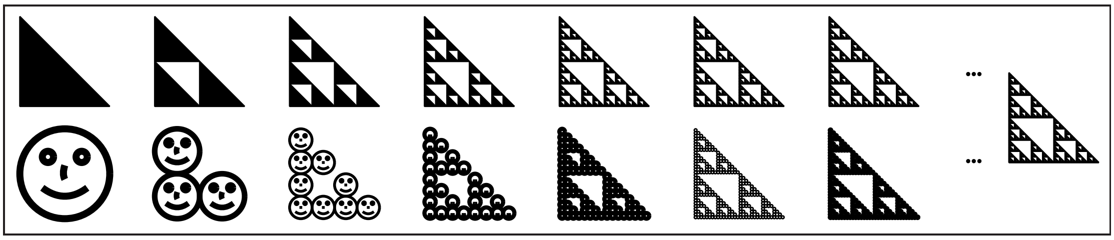


Figure 5: The Sierpinski triangle computed with two different "starting sets".

Definition 3.38 (totally disconnected IFS) *Let $\{(\mathbf{X}, d), \Omega\}$ be an IFS with attractor A and $\Omega = \omega_1, \ldots, \omega_n$. Then the IFS is said to be* totally disconnected, *if $\omega_i(A) \cap \omega_j(A) = \emptyset$ for all $i \neq j$.*

The Cantor set as introduced in the beginning of this section is the prototype of a fractal. Using the notion of iterated function systems we can re-define it as follows.

Definition 3.39 (Cantor set) *Let Ω be the set $\{\frac{1}{3}x; \frac{1}{3}x + \frac{2}{3}\}$ of contractive mappings on $\mathbb{R}$. Then the Cantor set is the attractor of the IFS $\{(\mathbb{R}, d_1), \Omega\}$.*

Definition 3.39 performs a direct embedding of Algorithm 3.2 into an iterated function system. The Cantor set is closed, bounded and hence compact, which is obvious since it is defined as an attractor of a hyperbolic IFS. Note that there is always a little gap between two points, but still the set is dense, i.e. in any arbitrarily small neighbourhood around any point of $\mathcal{C}$, there is some other point belonging to the Cantor set. Hence every point is an accumulation point and, therefore, it is also called a perfect set in topology. Instead of repeatedly removing the middle third of every piece, other fixed percentages ($0\% < x < 100\%$) could be used. The resulting sets are all homeomorphic to $\mathcal{C}$.

3.3.2. Fractal Dimensions

To complete this introduction into the theory of iterated function systems, fractal dimensions are briefly discussed in the sequel of this section. A fractal dimension can be understood as some kind of size of a fractal or how densely the space is covered by it. Figure 6 shows three similar structures, namely different versions of the Sierpinski triangle. Their fractal dimension is a way to grasp the similarity mathematically.

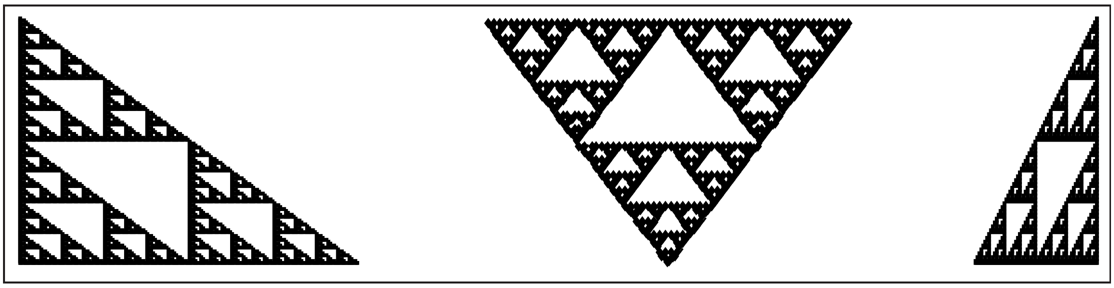


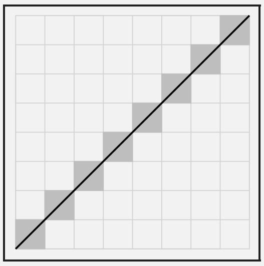
Figure 6: Three times the same fractal?

Definition 3.40 (fractal dimension) *Let* $(\mathbf{X}, d)$ *be a metric space and let* $A \in \mathcal{H}(\mathbf{X})$*. For each* $\epsilon > 0$ *let* $\mathcal{N}(A, \epsilon)$ *denote the number of closed balls of radius* ϵ *necessary to cover* A*. If* $D = \lim_{\epsilon \to 0} \frac{\ln(\mathcal{N}(A,\epsilon))}{\ln(1/\epsilon)}$ *exists, then* D *is called the* fractal dimension *of* A*.*

The following two theorems provide convenient ways to compute or measure the dimension of objects. The proofs of both can be found in [Bar93, p. 175, 183]. For "regular objects" the fractal dimension coincides with the usual notion of dimension, i.e. for example a line has a fractal dimension of 1 as shown in Example 3.42.

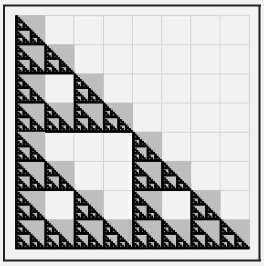
Theorem 3.41 (Box Counting Theorem) *Let* $(\mathbb{R}^n, d_n)$ *denote the* n*-dimensional Euclidean space. Cover the space completely using* n*-dimensional hyper-cubes of side length* $1/2^m$*. Let* $A \in \mathcal{H}(\mathbb{R}^n)$ *and let* $\mathcal{N}(A, m)$ *denote the number of hyper-boxes which intersect* A*. If* $D = \lim_{m \to \infty} \frac{\ln(\mathcal{N}(A,m))}{\ln(2^m)}$ *exists, then* D *is called the fractal dimension of* A*.*

Example 3.42 (The fractal dimension of a line and the Sierpinski triangle)



$$m = 3 \qquad \mathcal{N}(A,m) = 8 \qquad D_m = \frac{\ln(8)}{\ln(2^3)} = 1$$

$$l = 1/2^3 \qquad \mathcal{N}(A,m) = 27 \qquad D_m = \frac{\ln(27)}{\ln(2^3)} \approx 1.59$$



Theorem 3.43 *Let* $\{(\mathbf{X}, d), \Omega\}$ *be a totally disconnected hyperbolic IFS with attractor* A *and* $\Omega = \{\omega_1 \ldots \omega_n\}$*. Suppose* ω_i *is a similitude of scaling factor* s_n *for each* $\omega_i \in \Omega$*. Then* A *has the fractal dimension* D*, which is given by the unique solution of* $\sum_{i=1}^{n} s_i^D = 1$

Since the Cantor set is totally disconnected we can apply Theorem 3.43 to compute its dimension. The set is constructed using two similitudes with scaling factor $\frac{1}{3}$, hence $2 \cdot \left(\frac{1}{3}\right)^D = 1$. Therefore, we can compute the dimension D as $\frac{\ln 2}{\ln 3} \approx 0.63$. Some examples of fractals and their dimensions are shown in Figure 7.

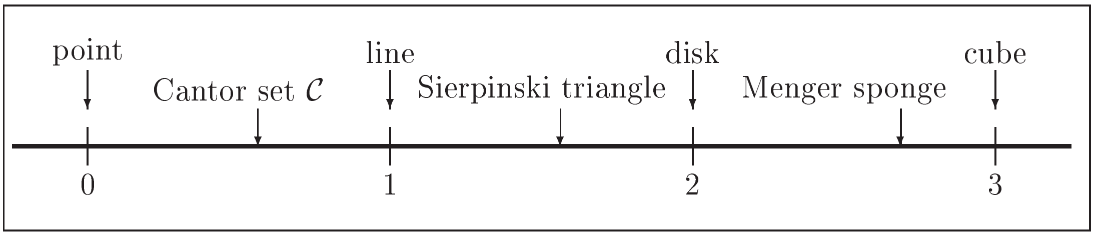


Figure 7: Dimensions of different objects

3.4. Computing the Attractor

Until now, the attractor is an purely mathematical infinite structure and it takes infinite time to compute it. This section will present two algorithms to compute an approximation in finite time, namely a deterministic and a stochastic version.

Algorithm 3.44 (Deterministic Computation of the Attractor)
Let $\{(\mathbf{X}, d), \Omega\}$ be a hyperbolic IFS with attractor A.

1. Choose a compact subset $A_0 \in \mathcal{H}(\mathbf{X})$.
2. Construct a sequence $A_0, A_1, A_2, \ldots$ with $A_{n+1} = \cup_{i=1}^{n} \omega_i(A_n)$.

According to Theorem 3.36 the constructed sequence converges to the attractor A of the IFS. As it is a so-called any-time-algorithm it can be interrupted after any step. After it is interrupted, one can simply plot the current attractor. This algorithm was used to create the images of the previous sections, e.g. Figure 4 shows the first eight steps of such a production. Unfortunately, this algorithm is very space consuming, as we need to store the attractor completely after each step. This is possible for affine transformations and rectangular starting sets, but usually very hard since any kind of mapping and any possible starting set is allowed.

The next algorithm follows a different approach. It creates a sequence of points of the attractor by applying randomly chosen mappings.

Algorithm 3.45 (Random Iteration of the Attractor)
Let $\{(\mathbf{X}, d), \Omega\}$ be a hyperbolic IFS with attractor A.

1. Attach a probability $p_i > 0$ to every mapping $\omega_i \in \Omega$, with $\sum p_i = 1$.
2. Choose a point $x_0 \in \mathbf{X}$.
3. Construct a sequence of points $x_0, x_1, x_2, \ldots$ with $x_{n+1} = \omega_i(x_n)$, by randomly choosing a mapping $\omega_i \in \Omega$ (according to p_i).

The proof of convergence for this algorithm is far beyond the scope of this thesis. It involves the theory of measures on fractals, which is presented in [Bar93, Chapter IX]. But even when the correctness is hard to prove the algorithm itself is fairly simple. It is easy to compute and very fast. To create an approximate image of the attractor, the sequence x_i is plotted. Appendix A shows pictures created using this algorithm.

In the sequel of this thesis an IFS will be illustrated the following way, where the black regions denote the image of the big rectangle.

$$\omega_1(x,y) = \begin{pmatrix} 0.5 & 0.0 \\ 0.0 & 0.5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$$

$$\omega_2(x,y) = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 0.25 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix}$$

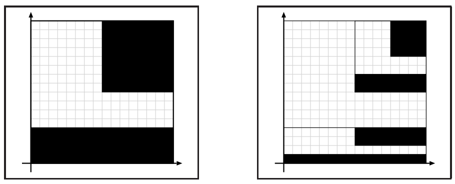


Figure 8: Illustration of an IFS as a collection of maps

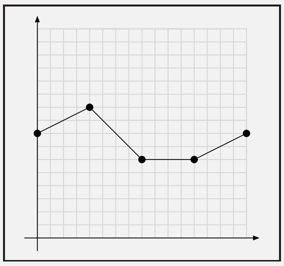
3.5. Fractal Interpolation

This section presents the idea of fractal interpolation as introduced in [Bar93]. An adapted version of the idea is used in Section 5.4, where the complete proofs are given. Therefore, proofs are omitted here and only the general idea is presented.

Interpolation tries to generalise a given set of data points. For that purpose simple functions are chosen to bridge the gap between the points and it is assumed that the chosen functions are good explanation for the missing data. Example 3.46 shows a set of data points and a corresponding linear interpolation.

Example 3.46 (piecewise linear interpolation)

$\mathbf{x_i}$	$\mathbf{y_i}$
0.0	0.5
0.25	0.75
0.5	0.25
0.75	0.25
1.0	0.5



To achieve good results a function must be chosen which suits the data, e.g. if a linear relation between the data is expected or known, linear interpolation functions are chosen. Usually “smooth” functions are used, since they fit our sense of simple.But this is strongly influenced by the Euclidean geometry, i.e. we expect things to be smooth.

Definition 3.47 (interpolation data) *A set of* interpolation data *is a sorted set of points* $\{(x_i, y_i) \mid 0 \leq i \leq n\}$ *with* $(x_i, y_i) \in \mathbb{R}^2$ *and* $x_i < x_j$ *for all* $i < j$.

Definition 3.48 (interpolation function) *Let* $D = \{(x_i, y_i) \mid 0 \leq i \leq n\}$ *be a set of interpolation data. Then a function* $f : [x_0, x_n] \to \mathbb{R}$ *is called an* interpolation function*, if it is continuous and* $f(x_i) = y_i$ *for all* $(x_i, y_i) \in D$.

Fractal interpolation tries to approximate “things outside the Euclidean geometry”, like attractors of iterated function systems. This section presents a method which constructs an IFS such that there is a continuous interpolation function for which the graph coincides with the attractor.

For this purpose we use iterated function systems $\{(\mathbb{R}^2, d_2), \Omega\}$, with $\Omega = \{\omega_1, \ldots, \omega_n\}$ consisting of “shear”-mappings of the form

$$\omega_i \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i & 0 \\ c_i & d_i \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix}.$$

Note that there is one mapping less than there are points in the data. In order to create an attractor which is the graph of a continuous function the following equations must hold for all the mappings:

$$\omega_i \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} x_{i-1} \\ y_{i-1} \end{pmatrix} \qquad \omega_i \begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$$

This leads to four equations which must hold, concerning the 5 variables (a_i, c_i, d_i, e_i and f_i). We chose d_i as a free parameter. This parameter controls the height of the shears. If it is set to $d_i = 0$ for all i, a linear interpolation is created. In Example 3.50 $d_i = 0.25$ is used, but it is not necessary that all the d_i are the same.

Algorithm 3.49 (Construction of interpolation "shear"-mappings)
Let $D = \{(x_i, y_i) \mid 0 \leq i \leq n\}$ be some interpolation data. This algorithm computes an IFS with n "shear"-mappings, for which the attractor passes through the $n+1$ points.

1. Choose a real number $0 \leq d_i < 1$ for each $1 \leq i \leq n$.
2. Compute the values for the other parameters according to:

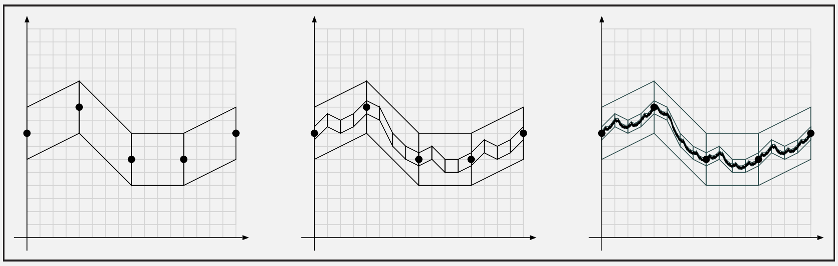
$$a_i = \frac{x_i - x_{i-1}}{x_n - x_0} \qquad e_i = \frac{x_n \cdot x_{i-1} - x_0 \cdot x_i}{x_n - x_0}$$

$$c_i = \frac{y_i - y_{i-1} - d_i \cdot (y_n - y_0)}{x_n - x_0} \qquad f_i = \frac{x_n \cdot y_{i-1} - x_0 \cdot y_i - d_i \cdot (x_n \cdot y_0 - x_0 \cdot y_n)}{x_n - x_0}$$

3. Return the set $\Omega\}$ with

$$\Omega = \{\omega_i \mid 1 \leq i \leq n\} \quad \text{and} \quad \omega_i \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i & 0 \\ c_i & d_i \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix}$$

Example 3.50 (fractal interpolation)



We have to ensure that all mappings of the IFS are contractions. Unfortunately the restriction $0 \leq d_i < 1$ is not sufficient to guarantee contractivity in $(\mathbb{R}^2, d_2)$, where d_2 denotes the 2-dimensional Euclidean metric. But it is sufficient with respect to the following "interpolation metric".

Definition 3.51 (interpolation metric d_i) *Let Ω be a set of mappings as constructed in Algorithm 3.49. Let θ be a real number with $\theta = 0$ if $c_i = 0$ for all $1 \leq i \leq n$ and otherwise $\theta = \frac{\min\{2-|a_n|\}}{\max\{2\cdot|c_n|\}}$. Then the* interpolation metric d_i *is defined as*

$$d_i\big((x_1, y_1), (x_2, y_2)\big) = |x_1 - x_2| + \theta \cdot |y_1 - y_2|.$$

Theorem 3.52 *Let $D = \{(x_i, y_i) \mid 0 < i \leq n\}$ be a set of interpolation data and Ω be a corresponding set of mappings as constructed in Algorithm 3.49. Let d_i be an appropriate interpolation metric. Then $(\mathbb{R}^2, d_i)$ is a complete metric space and the attractor of the hyperbolic IFS $\{(\mathbb{R}^2, d_i), \Omega\}$ is the graph of a continuous function $f : [x_0, x_n] \to \mathbb{R}$, with respect to the 1-dimensional Euclidean metric d_1. Furthermore, f interpolates the data, i.e. $f(x_i) = y_i$ for all $(x_i, y_i) \in D$.*

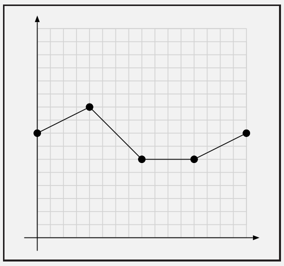
Definition 3.53 (fractal interpolation function) *The continuous function f from the theorem above is called a* fractal interpolation function *for the data.*

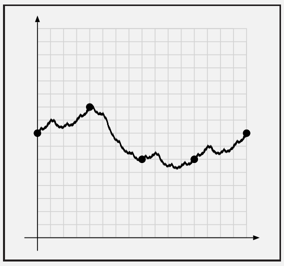
As mentioned above, a variant of this theorem can be found in Section 5.4 as Theorem 5.31, where a proof is given. Therefore, the proof is skipped here, but can be found in [Bar93, Chapter VI].

In Section 5.4 the ideas presented here are used to establish a connection between logic programs and a certain type of iterated function system. Furthermore, the study of fractal interpolation was motivated by the fact that the graphs of logic programs look fractal and hence should be approximated using methods from the fractal geometry. The following example depicts the different approaches. On the one hand a linear, on the other a fractal interpolation.

Example 3.54 (Comparing linear and fractal interpolation)

$\mathbf{x_i}$	$\mathbf{y_i}$
0.0	0.5
0.25	0.75
0.5	0.25
0.75	0.25
1.0	0.5





4. Graphs of Logic Programs

As mentioned in the introduction it is a desirable goal to combine the advantages of logic programs and artificial neural networks. But to achieve this we need to link the two different spaces these paradigms work on. Semantic operators of logic programs work on the space of interpretations as described in Section 2.2, whereas most types of artificial neural network work on n-dimensional real spaces.

Within this section a method is presented, which bridges the gap between the two different spaces. This method was developed by Hölldobler, Kalinke and Störr in [HKS99]. Some properties of the mapping are discussed and a real-valued version of the $T_{\mathcal{P}}$-operator is introduced, which leads to the definition of the *graph of a logic program*. The section finishes with a discussion of a generalised version of the Fitting metric and of properties of the graphs.

4.1. Mapping Interpretations to Real Numbers

First, we will identify the conditions, that must hold in order to link the space of interpretations and the real numbers in a meaningful way. Afterwards, one possible mapping is defined and its properties are discussed. Using this mapping the *right-shift operator* is introduced, which will help to prove and understand results achieved in Section 5.

4.1.1. Necessary Conditions for the Mappings

The embedding is defined as a function $f : I_{\mathcal{P}} \to \mathbb{R}$. As we need to distinguish the images $f(I)$ and $f(J)$ of two different interpretations I and J, we need that $f(I) \neq f(J)$, if $I \neq J$, hence the mapping must be injective.

From the motivation to approximate the graphs using a neural network, we need the range of the mapping to be a compact subset of $\mathbb{R}$ [Fun89]. But as mentioned in Section 3.3 this requirement is also necessary if we want to construct an iterated function system, that creates the same graph.

Therefore, we need the mapping to be injective and to map all elements of $I_{\mathcal{P}}$ into a compact subset of $\mathbb{R}$.

4.1.2. The Mapping by Hölldobler, Kalinke & Störr

The mapping presented here was defined by Hölldobler, Kalinke and Störr in [HKS99]. It maps interpretations to real numbers using an injective level-mapping. The intuition is to enumerate the Herbrand base according to the levels of the atoms. This enumeration of ground atoms is transferred into a list of 0-s and 1-s. Each atom that is mapped to *true*under I, is replaced with a “1” and the *false* ones with a “0”. This creates some kind of “binary” representation of the interpretation. If this list is interpreted as the number $0.list$ in an arbitrary number system (with base > 2), a unique real value $R(I)$ is assigned to each interpretation I.

Definition 4.1 (R) *Let $\mathcal{P}$ be a logic program, $\mathcal{B}_\mathcal{P}$ be its associated Herbrand base, $I_\mathcal{P}$ be the corresponding space of interpretations, $|\cdot| : \mathcal{B}_\mathcal{P} \to \mathbb{N}$ be an injective level mapping and let $B \in \mathbb{N}, B > 2$ be the base for the transformation. Then the mapping R assigns an unique real number $R(I)$ to every interpretation $I \in I_\mathcal{P}$ as follows:*

$$R : I_\mathcal{P} \to \mathbb{R} : I \mapsto \sum_{A \in I} B^{-|A|}$$

In the original definition the base B was set to 4 for various reasons (see [HKS99] for details). Throughout this thesis the base is left free as a parameter and only fixed for the examples. This creates a more general theory and allows to gain a deeper insight into the underlying mathematics.

Example 4.2 Let $\mathcal{L}(\mathcal{C}, \mathcal{F}, \mathcal{R}) = \mathcal{L}(\{0\}, \{s\}, \{e/1, o/1\})$ and assume the following level-mapping as used in Example 2.12:

$$|A| = \begin{cases} 2 \cdot n + 1 & \text{if } A = e(s^n(0)) \\ 2 \cdot n + 2 & \text{if } A = o(s^n(0)) \end{cases}$$

Let the base of the mapping R be $B = 4$.

$$\begin{aligned} I_2 &= \{e(0)\} \\ R(I_2) &= \sum_{A \in I} B^- |A| = 4^{-|e(0)|} = 4^{-1} = 0.25 \end{aligned}$$

$$\begin{aligned} I_1 &= \{e(0), o(s(0))\} \\ R(I_1) &= \sum_{A \in I} B^- |A| = 4^{-|e(0)|} + 4^{-|o(s(0))|} = 4^{-1} + 4^{-4} \approx 0.254 \end{aligned}$$

The following definition introduces the range of the mapping R. If the base of the mapping used is determined by the context the range will be denoted by D_f.

Definition 4.3 ($D_{\mathsf{f}B}$, R_m) *The range $D_{\mathsf{f}B} \subseteq \mathbb{R}$ of the mapping R is defined as $D_{\mathsf{f}B} = \{R(I) \mid I \in \mathcal{B}_\mathcal{P}\}$. The maximum value $R_\mathsf{m} = R(\mathcal{B}_\mathcal{P}) = \lim_{n\to\infty} \sum_{i=1}^{n} (B^{-i}) = \frac{1}{B-1}$.*

Remark 4.4 *For the sequel of this thesis we will treat R as a function from $I_\mathcal{P}$ to D_f, instead of $\mathbb{R}$.*

The probably most obvious base $B = 2$ does not create a valid mapping, since it is not possible to distinguish the numbers $0.0\bar{1}_2$ and 0.1_2, as their difference is 0. But for all $B > 2$ the mapping is injective, if the level-mapping is injective. As mentioned above, the injectivity is a required condition for a useful mapping.

Proposition 4.5 (injectivity of R) *Let R be the mapping constructed using the level-mapping $|\cdot|$ as introduced in Definition 4.1 and let the base B be a natural number larger then 2. Then R is injective, if the level mapping is injective.*

Proof The proof is done by contradiction. Let us assume that there are some $I \neq J$ such that $R(I) = R(J)$. From $I \neq J$ we know that there must be at an atom A, in which both interpretations differ. Without loss of generality we can assume that $A \in I$ and $A \notin J$ and that I and J agree on all atoms with level $< |A|$. Let $K = \{B \mid B \in I \text{ and } |B| < |A|\}$, i.e. the atom where I and J agree on and which have a level $< |A|$. Furthermore, let $I' = \{B \mid B \in I \text{ and } |B| > |A|\}$ and $J' = \{B \mid B \in J \text{ and } |B| > |A|\}$ denote the "tails" of I and J respectively. Using the definition of R it is easy to see, that

$$R(K) + R(\{A\}) \leq R(I) \leq R(K) + R(\{A\}) + R(I') \text{ and}$$
$$R(K) \leq R(J) \leq R(K) + R(J')$$

A upper bound r_{m} for $R(I')$ and $R(J')$ can be computed as follows:

$$\begin{aligned} r_{\mathsf{m}} &= \lim_{n\to\infty} \sum_{i=|A|+1}^{n} B^{-i} = \lim_{n\to\infty} \sum_{i=1}^{n} B^{-i} - \sum_{i=1}^{|A|} B^{-i} \\ &= \frac{1}{B-1} - \frac{1-B^{-|A|}}{B-1} = \frac{B^{-|A|}}{B-1} \end{aligned}$$

Therefore, we can conclude that

$$R(K) + B^{-|A|} \leq R(I) \leq R(K) + B^{-|A|} + \frac{B^{-|A|}}{B-1}$$
$$R(K) \leq R(J) \leq R(K) + \frac{B^{-|A|}}{B-1}$$

From the observation that $B^{-|A|} > \frac{B^{-|A|}}{B-1}$ for $B >$ we can conclude that both regions can never overlap and that for $I \neq J$ the images will always be different. Therefore, the mapping R is injective. A similar argument holds for R^{-1}. □

Corollary 4.6 *R is a bijection from $I_{\mathcal{P}}$ to D_{f}.*

Proposition 4.7 (continuity of R) *Let R be the mapping constructed using the injective level-mapping $|\cdot|$ and the base $B > 2$. Then R is a continuous mapping from $(I_{\mathcal{P}}, d_{\mathcal{P}B})$ to (D_{f}, d_1) and R^{-1} is a continuous mapping (D_{f}, d_1) to $(I_{\mathcal{P}}, d_{\mathcal{P}B})$.*

Proof We need to show that for all $\epsilon > 0$ there is a $\delta > 0$ such that for all $I, J \in I_{\mathcal{P}}$ we have that $d_{\mathcal{P}B}(I, J) < \delta$ implies that $d_1(R(I), R(J)) < \epsilon$. As in the proof of Proposition 4.5 we assume that I and J differ on some atom A with $|A| = n$ and agree on all atoms with level $< n$ and we assume that $A \in I$ and $A \notin J$. Furthermore, let K, I' and J' be defined as in the proof above. Then we have that $d_{\mathcal{P}B}(I, J) = B^{-n}$ and $d_1(R(I), R(J)) = B^{-n} + R(I') - R(J') \leq B^{-n} + \frac{B^{-n}}{B-1}$. Obviously there exists a n for each $\epsilon > 0$ such that $d_1(R(I), R(J)) \leq B^{-n} + \frac{B^{-n}}{B-1} < \epsilon$ for all $I, J \in I_{\mathcal{P}}$ with $d_{\mathcal{P}B}(I, J) < B^{-n} = \delta$. Therefore, we can conclude that R is continuous. A similar argument holds for R^{-1}. $\square$

Corollary 4.8 *R is a homeomorphism from $I_{\mathcal{P}}$ to $D_{\mathsf{f}B}$.*

Corollary 4.9 *$D_{\mathsf{f}i}$ and $D_{\mathsf{f}j}$ are homeomorphic for all $i, j > 2$.*

There is also a strong relation between $(D_{\mathsf{f}B}, d_1)$ and $(\mathcal{C}, d_1)$, since the Cantor set $\mathcal{C}$ is obviously a scaled version of $D_{\mathsf{f}3}$. Therefore, $(D_{\mathsf{f}B}, d_1)$ and $(\mathcal{C}, d_1)$ are homeomorphic for all $B > 2$.

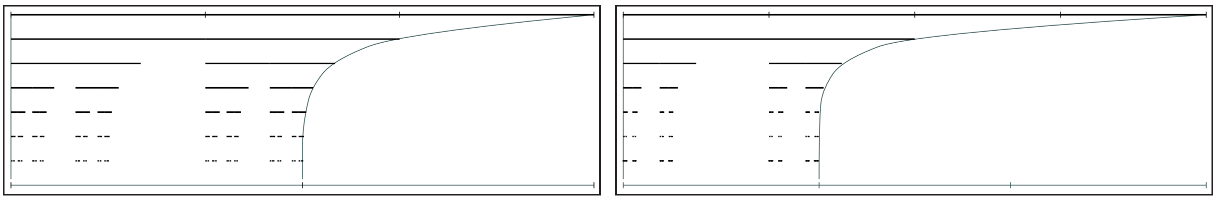


Figure 9: The similarity of $D_{\mathsf{f}3}$ and $D_{\mathsf{f}4}$

As mentioned above, it is necessary that the image of $I_{\mathcal{P}}$ is a compact subset of $\mathbb{R}$. This is required by the original motivation to link logic programs and neural networks, as well as from the motivation to link logic programs and iterated function systems.

Proposition 4.10 *$D_{\mathsf{f}B}$ is a compact subset of $\mathbb{R}$.*

Proof (sketch) As there is a continuous map ($D_{\mathsf{f}3} = \{2 \cdot x \mid x \in \mathcal{C}\}$) from $\mathcal{C}$ to $D_{\mathsf{f}3}$, $D_{\mathsf{f}3}$ must be compact, hence $D_{\mathsf{f}B}$ in general must be compact. $\square$

Proof (sketch) Using the fact that $(I_{\mathcal{P}}, d_{\mathcal{P}B})$ is a complete metric space (see Proposition 4.21) it is easy to show that every Cauchy sequence $x_1, x_2, \ldots$ with $x_i \in D_{\mathsf{f}B}$ for all i converges to an element of $D_{\mathsf{f}B}$, which turns $(D_{\mathsf{f}B}, d_1)$ into a complete metric space. It is obvious that $D_{\mathsf{f}B}$ is bounded. Finally the Heine-Borel theorem tells us that every closed and bounded subset of $\mathbb{R}$ is compact, which ends the proof. $\square$

Corollary 4.11 *Let d_1 be the 1-dimensional Euclidean metric. Then $(D_{\mathsf{f}B}, d_1)$ is a complete metric space.*

To simplify the proofs in the sequel we want to introduce and prove some equations that hold for R in general.

Proposition 4.12 *Let R be the mapping introduced in Definition 4.1. Then the following equations hold for all $i, j \in D_f$ and all $I, J \in \mathcal{B}_{\mathcal{P}}$:*

$$
\begin{aligned}
R(R^{-1}(i)) &= i && && (\mathbf{R_1})\\
R^{-1}(R(I)) &= I && && (\mathbf{R_2})\\
R(I \cup J) &= R(I) + R(J) - R(I \cap J) && && (\mathbf{R}_\cup)\\
R(I \setminus J) &= R(I) - R(J) && \textit{iff } J \subseteq I && (\mathbf{R}_\setminus)\\
R^{-1}(i + j) &= R^{-1}(i) \cup R^{-1}(j) && \textit{iff } R^{-1}(i) \cap R^{-1}(j) = \emptyset && (\mathbf{R}_+)
\end{aligned}
$$

Proof The property $(\mathbf{R_1})$ holds for all functions and $(\mathbf{R_2})$ for all injective functions in general. Note that set union is not invertible even if the result and one part is known. Therefore, some atoms might be counted twice, when a set is split into two parts. Using the definition of R we get $(\mathbf{R}_\cup)$ as follows:

$$
\begin{aligned}
R(I \cup J) = \sum_{A \in I \cup J} B^{-|A|} &= \sum_{A \in I} B^{-|A|} + \sum_{A \in J} B^{-|A|} - \sum_{A \in I \cap J} B^{-|A|}\\
&= R(I) + R(J) - R(I \cap J)
\end{aligned}
$$

To prove $(\mathbf{R}_\setminus)$ and using $J \subseteq I$ we get

$$
\begin{gathered}
R(I) = R((I \setminus J) \cup J) = R(I \setminus J) + R(J) - R((I \setminus J) \cap J)\\
\text{which can be rewritten as}\\
R(I \setminus J) = R(I) - R(J) \text{ since } (I \setminus J) \cap J = \emptyset \text{ and } R(\emptyset) = 0
\end{gathered}
$$

In order to show $(\mathbf{R}_+)$ we first need to show that $i + j \in D_f$ iff $i, j \in D_f$ and $R^{-1}(i) \cap R^{-1}(j) = \emptyset$. Since $i, j \in D_f$ we know that there are two interpretations I and J with $R(I) = i$, $R(J) = j$ and $I \cap J = \emptyset$. Using $(\mathbf{R}_\cup)$ and $I \cup J \in I_{\mathcal{P}}$ we get that $i + j = R(I \cup J) \in D_f$. Knowing that R is injective it is sufficient to show that $R(R^{-1}(i + j)) = R(R^{-1}(i) \cup R^{-1}(j))$.

$$
\begin{aligned}
R(R^{-1}(i + j)) = i + j &= R(R^{-1}(i)) + R(R^{-1}(j))\\
&\text{knowing that } R^{-1}(i) \cap R^{-1}(j) = \emptyset \text{ we can conclude}\\
&= R(R^{-1}(i)) + R(R^{-1}(j)) - R(R^{-1}(i) \cap R^{-1}(j))\\
&\text{using the equation } \mathbf{R}_\cup \text{ we get}\\
&= R(R^{-1}(i) \cup R^{-1}(j))
\end{aligned}
$$

□

4.1.3. The Right-shift operator

The following operator will help to prove and understand results achieved later in this thesis. It uses the mapping R from the previous section to define a shift operation on the space of interpretations.

Definition 4.13 (right-shift operator) *Let R be defined as in Definition 4.1 using the level-mapping $|\cdot|$ and the base B. Then the* right-shift operator $\overset{\cdot}{\underset{\cdot}{\rightarrow}}$ *is defined as follows:*

$$\overset{\cdot}{\underset{\cdot}{\rightarrow}} : I_{\mathcal{P}} \times \mathbb{N} \rightarrow I_{\mathcal{P}} : \overset{I}{\underset{P}{\rightarrow}} = R^{-1}\left(\frac{R(I)}{B^P}\right)$$

Assume the Herbrand base sorted according to the level-mapping $|\cdot|$. If every atom in this sequence is replaced with a "1" if it is contained in the interpretation I, with a "0" otherwise, you get a "binary" representation of I. This sequence is shifted to the right by adding P zeros in the beginning, hence the name. The resulting interpretation $\overset{I}{\underset{P}{\rightarrow}}$ is the shifted sequence transferred into the corresponding set of atoms.

Proposition 4.14 *Let $\overset{\cdot}{\underset{\cdot}{\rightarrow}}$ be defined as above, let I be an interpretation and P be a natural number. Then $\overset{I}{\underset{P}{\rightarrow}} = \{A \mid \exists A' \in I \text{ with } |A'| + P = |A|\}$ holds for all $I \in I_{\mathcal{P}}$ and all $P \in \mathbb{N}$.*

Proof The equation follows immediately from the definition of R.

$$\overset{I}{\underset{P}{\rightarrow}} = R^{-1}\left(\frac{R(I)}{B^P}\right) = R^{-1}\left(\frac{\sum_{A\in I} B^{-|A|}}{B^P}\right) = R^{-1}\left(\sum_{A\in I} B^{-(|A|+P)}\right)$$

Therefore, we can conclude $\overset{I}{\underset{P}{\rightarrow}} = \{A \mid \exists A' \in I \text{ with } |A'| + P = |A|\}$. $\square$

Example 4.15 Let $\mathcal{L}(\mathcal{C}, \mathcal{F}, \mathcal{R}) = \mathcal{L}(\{0\}, \{s\}, \{e/1, o/1\})$ and assume the following level-mapping:

$$|A| = \begin{cases} 2 \cdot n + 1 & \text{if } A \text{ is of the form } e(s^n(0)) \\ 2 \cdot n + 2 & \text{if } A \text{ is of the form } o(s^n(0)) \end{cases}$$

Therefore, the Herbrand base $\mathcal{B}_{\mathcal{P}}$ can be written as the sorted list

$$l(\mathcal{B}_{\mathcal{P}}) = [e(0), o(0), e(s(0)), o(s(0)), e(s(s(0))), o(s(s(0))), \ldots]$$

$$\begin{aligned} I &= \{e(0), o(s(0))\} & R(I) &= B^{-1} + B^{-4} \\ l(I) &= [1, 0, 0, 1, 0, 0, \ldots] & & \\ \overset{l(I)}{\underset{2}{\rightarrow}} &= [0, 0, 1, 0, 0, 1, 0, 0, \ldots] & & \\ \overset{I}{\underset{2}{\rightarrow}} &= \{e(s(0)), o(s(s(0)))\} & R(\overset{I}{\underset{2}{\rightarrow}}) &= B^{-3} + B^{-6} \end{aligned}$$

Analogous to the right-shift operator a left-shift operator $\overset{I}{\underset{P}{\leftarrow}}$ can be defined. For this operator it is necessary that the atoms with level smaller or equal to P are not contained in I, otherwise the result would not be an interpretation.

4.2. Mapping Programs to Graphs in the Real Numbers

The mapping R as defined in the previous section can be used to embed the $T_{\mathcal{P}}$-operator into the real numbers. As described in Section 2.2.2 this operator maps an interpretation I to its immediate consequence $T_{\mathcal{P}}(I)$. Using the mapping R we can embed $T_{\mathcal{P}}$ into $\mathbb{R}$ as shown in Figure 10.

$$\begin{array}{ccc} I \in I_{\mathcal{P}} & \xrightarrow{T_{\mathcal{P}}} & I' \in I_{\mathcal{P}} \\ R\big\downarrow\big\uparrow R^{-1} & & R\big\downarrow\big\uparrow R^{-1} \\ i \in D_{\mathsf{f}} \subset \mathbb{R} & \xrightarrow{f_{\mathcal{P}}} & i' \in D_{\mathsf{f}} \subset \mathbb{R} \end{array}$$

Figure 10: The relation between $T_{\mathcal{P}}$ and $f_{\mathcal{P}}$

Using $T_{\mathcal{P}}$ and R we can define a function $f_{\mathcal{P}}$ which is a real-valued version of $T_{\mathcal{P}}$. The graph of this mapping can be plotted as a subset of $\mathbb{R}^2$.

Definition 4.16 (the embedded $T_{\mathcal{P}}$-operator $f_{\mathcal{P}}$) *Let $\mathcal{P}$ be a logic program, $T_{\mathcal{P}}$ its associated consequence operator and let $R : I_{\mathcal{P}} \to D_f$ be the mapping as defined above. Then the function $f_{\mathcal{P}}$ is defined as follows:*

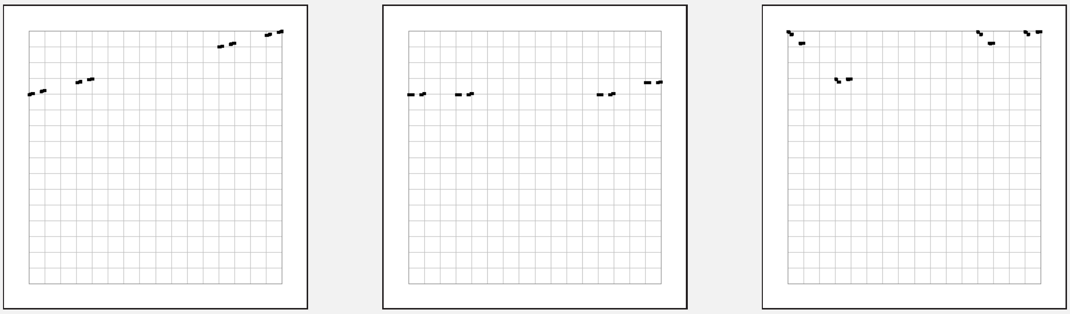
$$f_{\mathcal{P}} : D_f \to D_f : r \mapsto R(T_{\mathcal{P}}(R^{-1}(r)))$$

Remark 4.17 *The operator $f_{\mathcal{P}}$ is denoted $\bar{f}_{\mathcal{P}}$ in [HKS99].*

Definition 4.18 ($F_{\mathcal{P}}$) *Let $\mathcal{P}$ be a logic program, $T_{\mathcal{P}}$ its associated consequence operator and $f_{\mathcal{P}}$ be the emdedded version. Then the graph $F_{\mathcal{P}}$ is defined as*

$$F_{\mathcal{P}} = \{(x, y) \mid x = R(I), y = R(T_{\mathcal{P}}(I)), I \in \mathcal{B}_{\mathcal{P}}\} = \{(x, f_{\mathcal{P}}(x)) \mid x \in D_f\}.$$

Example 4.19 (Graphs of logic programs)



Some more examples of logic programs and their graphs can be found in appendix B. Note that this graph contains all consequence relations $(I, T_{\mathcal{P}}(I))$ as defined by the $T_{\mathcal{P}}$-operator, i.e. everything contained in the program $\mathcal{P}$ with respect to the immediate consequence operator.

4.3. A Generalised Fitting Metric

In this section a generalised version of the Fitting metric is introduced. The original version $d_{\mathcal{P}}$ was first described in [Fit94]. This generalised version is introduced, since it gives deeper insights into connection between the space of interpretations and D_{f}. It allows to simplify proofs done in [HKS99, McI00] concerning the contractivity of the embedded $T_{\mathcal{P}}$-operator for acyclic logic programs.

Definition 4.20 ($d_{\mathcal{P}B}$) *Let $\mathcal{B}_{\mathcal{P}}$ be a Herbrand base and $I_{\mathcal{P}}$ be the corresponding space of interpretations. Let $|\cdot|$ be an injective level-mapping and R be defined as above using $|\cdot|$ and the base B. Then the function $d_{\mathcal{P}B} : I_{\mathcal{P}} \times I_{\mathcal{P}} \to \mathbb{R}$ is defined as follows:*

$$d_{\mathcal{P}B}(I,J) = \begin{cases} 0 & \text{if } I = J \\ B^{-|A|} & \text{if } I \text{ and } J \text{ differ on the atom } A, \\ & \text{but agree on all atoms with a level smaller than } |A|. \end{cases}$$

For $B = 2$, $d_{\mathcal{P}B}$ coincides with the usual Fitting metric $d_{\mathcal{P}}$, as introduced in [Fit94].

Proposition 4.21 (the space $(I_{\mathcal{P}}, d_{\mathcal{P}B})$) *Let $d_{\mathcal{P}B}$ be defined as above. Then $(I_{\mathcal{P}}, d_{\mathcal{P}B})$ is a complete metric space.*

Proof (sketch) Since R^{-1} is a continuous bijection and (D_{f}, d_1) is compact, $(I_{\mathcal{P}}, d_{\mathcal{P}B})$ must be compact, hence complete. □

Proof First, we will show that $d_{\mathcal{P}B}$ is a metric on $I_{\mathcal{P}}$ and, afterwards, that the space is complete. To show that $d_{\mathcal{P}B}$ is a metric we need to show that all three conditions from Definition 3.3 hold for all $I, J, K \in I_{\mathcal{P}}$. **(i)** $d_{\mathcal{P}B}(I,I) = 0$ and **(ii)** $d_{\mathcal{P}B}(I,J) = d_{\mathcal{P}B}(I,J)$ follow immediately from the definition of $d_{\mathcal{P}B}$. To show **(iii)** $d_{\mathcal{P}B}(I,K) \leq d_{\mathcal{P}B}(I,J) + d_{\mathcal{P}B}(J,K)$, we assume without loss of generality that $d_{\mathcal{P}B}(I,J) = B^{-m}$, $d_{\mathcal{P}B}(J,K) = B^{-n}$ and that $m < n$. Since I and J agree on all atoms with level smaller than m and $m < n$, we know that J and K must agree on those atoms as well. Therefore, I and K must agree on all those atoms and consequently $d_{\mathcal{P}B}(I,K) = B^{-m}$. Therefore, $d_{\mathcal{P}B}(I,K) \leq \max(d_{\mathcal{P}B}(I,J), d_{\mathcal{P}B}(J,K))$ and **(iii)** holds.
To prove that $(I_{\mathcal{P}}, d_{\mathcal{P}B})$ is a complete metric space, we need to show that every Cauchy sequence $I_1, I_2, I_3, \ldots$ converges to an $I \in I_{\mathcal{P}}$. Since the sequence is Cauchy it follows that for every $\epsilon > 0$ there is an $N \in \mathbb{N}$ such that for all $i, j > N$ we have $d_{\mathcal{P}B}(I_i, I_j) < \epsilon$. I.e. for each ϵ there is an $n \in \mathbb{N}$, such that $B^{-n} \leq \epsilon$ and I and J agree on all atom with a level smaller than n. Therefore, we can conclude that for any atom A with $|A| = n$ there is an N, such that all interpretations I_l with $l > N$ agree on A, this value of A is denoted A'. The interpretation I is defined as follows. For every $A \in \mathcal{B}_{\mathcal{P}}$ we assume that I contains A, if and only if A' is true. Therefore, we can conclude that for every $\epsilon > 0$ there is an $N \in \mathbb{N}$, such that for all $n < N$ we have $d_{\mathcal{P}B}(I_n, I) < \epsilon$, what shows that $I \in I_{\mathcal{P}}$ is the limit point of the sequence. □

Remark 4.22 *Note that $d_{\mathcal{P}B}(I,K) \leq \max(d_{\mathcal{P}B}(I,J), d_{\mathcal{P}B}(J,K))$ holds which turns $d_{\mathcal{P}B}$ into an* ultra metric *on $I_{\mathcal{P}}$.*

The following Theorem was first shown by Fitting in [Fit94]. He showed that the $T_{\mathcal{P}}$-operator associated with an acyclic logic programs is a contraction on the complete metric space $(I_{\mathcal{P}}, d_{\mathcal{P}})$. This allows to conclude that there must be a fixed point of the operator as shown by Banach. Here the result is generalised with respect to the base. Afterwards a theorem is given which establishes a strong connection between the complete metric space of interpretations and the 1-dimensional Euclidean space.

Theorem 4.23 *Let $\mathcal{P}$ be an acyclic logic program with respect to the level mapping $|\cdot|$. Then $T_{\mathcal{P}}$ is a contraction on $(I_{\mathcal{P}}, d_{\mathcal{P}B})$ for each $B > 2$ with contractivity factor B^{-1}.*

Proof In order to show that $T_{\mathcal{P}}$ is a contraction with the specified contractivity factor, we need to show that $d_{\mathcal{P}B}(T_{\mathcal{P}}(I), T_{\mathcal{P}}(J)) \leq \frac{1}{B} \cdot d_{\mathcal{P}B}(I, J)$ holds for all interpretations $I, J \in I_{\mathcal{P}}$. This corresponds to $B^{-m} \leq \frac{1}{B} \cdot B^{-n}$, where m is the smallest level of an atom where $T_{\mathcal{P}}(I)$ and $T_{\mathcal{P}}(J)$ differ in and respectively n for I and J.
If $T_{\mathcal{P}}(I) = T_{\mathcal{P}}(J)$ this condition is satisfied. Without loss of generality we assume that $T_{\mathcal{P}}(I)$ and $T_{\mathcal{P}}(J)$ differ in the atom A with level m and agree on all atoms with a level smaller than m, $A \in T_{\mathcal{P}}(I)$ and $A \notin T_{\mathcal{P}}(J)$.
From $T_{\mathcal{P}}(I) \neq T_{\mathcal{P}}(J)$ and $A \in T_{\mathcal{P}}(I)$ we conclude that there must be at least one clause $A \leftarrow L_1, \ldots, L_k$ in $\mathcal{P}$ where $I \models L_1 \wedge \ldots \wedge L_k$ and $J \not\models L_1 \wedge \ldots \wedge L_k$. Therefore, there must be an $L_i \in \{L_1, \ldots, L_k\}$ with $I \models L_i$ and $J \not\models L_i$. From Definition 2.13 and the fact that $\mathcal{P}$ is acyclic it follows that $|A| > |L_i|$ holds for all $L_i \in \{L_1, \ldots, L_k\}$. Therefore, we can conclude that if $T_{\mathcal{P}}(I)$ and $T_{\mathcal{P}}(J)$ differ in an atom with level m, than there is an atom A_i with $m > |A_i|$, such that I and J differ in A_i. This finishes the proof, because we can conclude that $B^{-m} \leq B^{-(n+1)}$, where $|A_i| = n$ and hence $d_{\mathcal{P}B}(T_{\mathcal{P}}(I), T_{\mathcal{P}}(J)) \leq \frac{1}{B} \cdot d_{\mathcal{P}B}(I, J)$. □

Theorem 4.24 *Let $\mathcal{B}_{\mathcal{P}}$ be a Herbrand base and $I_{\mathcal{P}}$ be the corresponding space of interpretations. Let $|\cdot|$ be an injective level-mapping and let R and $d_{\mathcal{P}B}$ be defined as above using $|\cdot|$ and the base B. Then the two complete metric spaces $(I_{\mathcal{P}}, d_{\mathcal{P}B})$ and (D_{f}, d_1) are equivalent as introduced in Definition 3.6, connected by $c_1 = \frac{B-2}{B-1}$ and $c_2 = \frac{B}{B-1}$.*

Proof Let $\tilde{d}_{\mathcal{P}B}(I, J) = d_1(R(I), R(J))$. In order to prove the equivalence of $(I_{\mathcal{P}}, d_{\mathcal{P}B})$ and (D_{f}, d_1) it suffices to show that for all $I, J \in I_{\mathcal{P}}$ we have that $c_1 \cdot d_{\mathcal{P}B}(I, J) \leq \tilde{d}_{\mathcal{P}B}(I, J) \leq c_2 \cdot d_{\mathcal{P}B}(I, J)$ holds. Without loss of generality we assume that I and J disagree on some atom A and agree on all atoms with level $< |A|$, $A \in I$ and $A \notin J$. As in the proof of Proposition 4.5 let $K = \{B \mid B \in I \text{ and } |B| < |A|\}$ and I' and J' denote the "tails" of I and J respectively, i.e. $I = K \cup \{A\} \cup I'$ and $J = K \cup J'$.
Using the definition of $\tilde{d}_{\mathcal{P}B}$ and R we get $\tilde{d}_{\mathcal{P}B}(I, J) = R(I) - R(J)$, because $R(I) > R(J)$ what can be easily verified (see Proposition 4.5). This can be rewritten as $\tilde{d}_{\mathcal{P}B}(I, J) = \big(R(K) + B^{-|A|} + R(I')\big) - (R(J) + R(J')) = B^{-|A|} + R(I') - R(J')$. As shown in the proof of Proposition 4.5, the upper bound for $R(I')$ and $R(J')$ is $\frac{B^{-|A|}}{B-1}$. Therefore, we can conclude that $B^{-|A|} - \frac{B^{-|A|}}{B-1} \leq \tilde{d}_{\mathcal{P}B}(I, J) \leq B^{-|A|} + \frac{B^{-|A|}}{B-1}$, which can be rewritten as $c_1 \cdot d_{\mathcal{P}B}(I, J) \leq \tilde{d}_{\mathcal{P}B}(I, J) \leq c_2 \cdot d_{\mathcal{P}B}(I, J)$. □

Lemma 4.25 *The bounds c_1 and c_2 from Theorem 4.24 are optimal, i.e. there is no greater c_1 and no smaller c_2 such that the metric spaces $(I_\mathcal{P}, d_{\mathcal{P}B})$ and (D_f, d_1) are equivalent.*

Proof To show that $c_1 = \frac{B-2}{B-1}$ is optimal we will show that there is no $\epsilon > 0$ such that $\left(\frac{B-2}{B-1} + \epsilon\right) \cdot d_{\mathcal{P}B}(I, J) \leq d_1(R(I), R(J))$. Without loss of generality we assume that I and J disagree on some atom A with $|A| = n$ and agree on all atoms with level $< n$, $A \in I$ and $A \notin J$. Using the notion from the proof of Theorem 4.24 we can conclude that $d_1(R(I), R(J)) = B^{-n} + R(I') - R(J')$, which is minimal with $I' = \emptyset$ and $J' = \{A \mid |A| > n\}$. Since $\left(\frac{B-2}{B-1} + \epsilon\right) \cdot d_{\mathcal{P}B}(I, J) \leq d_1(R(I), R(J))$ must hold for all $I, J \in I_\mathcal{P}$ we have that $\left(\frac{B-2}{B-1} + \epsilon\right) \cdot B^{-n} \leq B^{-n} + R(\emptyset) - R(J')$ and therefore $\left(\frac{B-2}{B-1} + \epsilon\right) \cdot B^{-n} \leq B^{-n} - \frac{B^{-n}}{B-1}$ must hold. This gives $\frac{B-2}{B-1} + \epsilon \leq 1 - \frac{1}{B-1}$ and we can conclude that $\epsilon = 0$. A similar argument holds for c_2 and, therefore, we can conclude that both bounds are optimal. □

Proposition 4.26 *Let $\mathcal{P}$ be an acyclic logic program with respect to some level mapping $|\cdot|$ and let $T_\mathcal{P}$ be the associated consequence operator. Let R be the embedding from Definition 4.1 using base $B > 3$. Then the embedded operator $f_\mathcal{P}$ is a contraction on (D_f, d_1) with contractivity factor $s = \frac{1}{B-2}$, where d_1 denotes the 1-dimensional Euclidean metric.*

Proof From Theorem 4.24 above we know that $(I_\mathcal{P}, d_{\mathcal{P}B})$ and (D_f, d_1) are equivalent, using the mapping R and the two constants $c_1 = \frac{B-2}{B-1}$ and $c_2 = \frac{B}{B-1}$. Knowing that $f_\mathcal{P}$ is a contraction on $(I_\mathcal{P}, d_{\mathcal{P}B})$ with contractivity factor $\frac{1}{B}$ (see Proposition 4.23) and using Proposition 3.30 we know that $f_\mathcal{P}$ is a contraction if $0 \leq \frac{c_2 \cdot s_f}{c_1} < 1$. Therefore, it is sufficient to show that the latter inequality holds. Using the given values we derive $0 \leq \frac{\frac{B}{B-1} \cdot \frac{1}{B}}{\frac{B-2}{B-1}} = \frac{1}{B-2} < 1$, what holds for all $B > 3$. □

Corollary 4.27 *Let $\mathcal{P}$ be an acyclic logic program and $f_\mathcal{P}$ its embedded $T_\mathcal{P}$-operator. Then $f_\mathcal{P}$ is continuous on (D_f, d_1).*

Note that the base $B = 3$ does not ensure the contractivity of $f_\mathcal{P}$.

5. From Logic Programs to Iterated Function Systems

As seen in the previous sections, logic programs (using the embedding R) and iterated function systems generate similar graphs. In this section conditions are identified under which both graphs coincide. First, the general structure of the iterated function systems is defined and theorems are given, which establish connections between iterated function systems and logic programs. Afterwards, the linear case is discussed and a method proposed to construct an IFS for a given program $\mathcal{P}$. For this IFS it is shown that the attractor A coincides with the graph $F_\mathcal{P}$ of the logic program, if certain conditions are met. Furthermore, the idea of fractal interpolation is used to find approximations for the graphs of logic programs.

5.1. The General Structure and the Equivalence Theorem

As mentioned above, logic programs and iterated function systems create similar graphs. This section links both by giving necessary and sufficient conditions under which the graph $F_\mathcal{P}$ of a logic program is the attractor of a hyperbolic iterated function system. Since the iterated function systems are intended to approximate graphs in $\mathbb{R}^2$, they must be defined on that space. Therefore, we will use the space $(\mathbb{R}^2, d_2)$, where d_2 denotes the 2-dimensional Euclidean metric. This space is well known, e.g. it is known to be a complete metric space, as shown in Proposition 3.23.

Definition 5.1 ($IFS_\mathcal{P}$) *An iterated function system $IFS_\mathcal{P}$ associated with a logic program $\mathcal{P}$ is defined as follows:*

$$IFS_\mathcal{P} = \{(\mathbb{R}^2, d_2), \Omega\} \qquad\qquad \Omega = \{\omega_1, \dots, \omega_n\},$$

with ω_i being a composition of two mappings

$$\omega_i : \mathbb{R}^2 \to \mathbb{R}^2 : (x, y) \mapsto \left(\omega_i^{\mathsf{x}}(x, y), \omega_i^{\mathsf{y}}(x, y)\right)$$

where d_2 denotes the 2-dimensional Euclidean metric as introduced in Definition 3.22 and each ω_i is a contractive mapping on $(\mathbb{R}^2, d_2)$.

The definition above simply restricts the IFSs to 2-dimensional real ones. So far we do not restrict the mappings to a certain type, as long as they are contractive. As shown in Section 3.3 this ensures that there is a uniquely determined attractor. The following theorem provides the promised conditions for the coincidence of the graph $F_\mathcal{P}$ and the attractor of an $IFS_\mathcal{P}$.

As mentioned in the Introduction the small x and y are used to indicate the corresponding coordinates. For example the mapping ω_i^{x} is not indexed by x, but rather denotes a function on the x-coordinate.

Theorem 5.2 (Equivalence Theorem) *Let $\mathcal{P}$ be a logic program, $f_\mathcal{P}$ be the embedded $T_\mathcal{P}$-operator, let $F_\mathcal{P}$ be its graph and D_f be the range of the mapping R, as introduced in Section 4. Let $IFS_\mathcal{P} = \{(\mathbb{R}^2, d_2), \Omega\}$ be a (hyperbolic) iterated function system as introduced in Definition 5.1 and let A be its uniquely determined attractor. Furthermore, let $\pi_\mathsf{x}(\cdot)$ denote the projection to the x-axis. Then the graph $F_\mathcal{P}$ coincides with the attractor A, i.e. $F_\mathcal{P} = A$, if and only if $\pi_\mathsf{x}(A) = D_\mathsf{f}$ and $f_\mathcal{P}(\omega_i^\mathsf{x}(a)) = \omega_i^\mathsf{y}(a)$ holds for all $a \in F_\mathcal{P}$ and all $\omega_i \in \Omega$.*

Proof The proof is split into two parts. First, we will show that $F_\mathcal{P} = A$ if $f_\mathcal{P}(\omega_i^\mathsf{x}(a)) = \omega_i^\mathsf{y}(a)$ and $\pi_\mathsf{x}(A) = D_\mathsf{f}$ and, afterwards, the converse.

(i) To show the equivalence of $F_\mathcal{P}$ and A, we need to show that $F_\mathcal{P} \subseteq A$ and $A \subseteq F_\mathcal{P}$. **(i.a)** From $\Omega(A) = A$ and $\pi_\mathsf{x}(A) = \pi_\mathsf{x}(F_\mathcal{P}) = D_\mathsf{f}$ we know that for each $a \in F_\mathcal{P}$ there must be an $a' \in A$ and an $\omega_i \in \Omega$, such that $\pi_\mathsf{x}(a) = \omega_i^\mathsf{x}(a')$. Using $f_\mathcal{P}(\omega_i^\mathsf{x}(a')) = \omega_i^\mathsf{y}(a')$ and the definition of $f_\mathcal{P}$ we know that $\omega_i^\mathsf{y}(a') = f_\mathcal{P}(\pi_\mathsf{x}(a)) = \pi_\mathsf{y}(a)$. Therefore, we can conclude that $(\pi_\mathsf{x}(a), \pi_\mathsf{y}(a)) = (\omega_i^\mathsf{x}(a'), \omega_i^\mathsf{y}(a'))$, i.e. $a = \omega_i(a')$. Since $a' \in A$, hence $\omega_i(a') \in A$, we can conclude that $a \in A$ and finally $F_\mathcal{P} \subseteq A$. **(i.b)** From $f_\mathcal{P}(\omega_i^\mathsf{x}(a)) = \omega_i^\mathsf{y}(a)$ and Definition 4.18 we can conclude that $(\omega_i^\mathsf{x}(a), \omega_i^\mathsf{y}(a)) \in F_\mathcal{P}$, i.e. $\omega_i(a) \in F_\mathcal{P}$. Knowing that this equation holds for all $a \in F_\mathcal{P}$ and all $\omega_i \in \Omega$ we get $\omega_i(F_\mathcal{P}) \subseteq F_\mathcal{P}$ and finally $\Omega(F_\mathcal{P}) \subseteq F_\mathcal{P}$. Therefore, we can conclude that $F_\mathcal{P} = A$.

(ii) Since $F_\mathcal{P} = A$ and $\pi_\mathsf{x}(F_\mathcal{P}) = D_\mathsf{f}$ we get $\pi_\mathsf{x}(A) = D_\mathsf{f}$ immediately. Furthermore, we know that $F_\mathcal{P} = \Omega(F_\mathcal{P})$ and hence $\Omega(a) \subseteq F_\mathcal{P}$ holds for all $a \in F_\mathcal{P}$. Therefore, we can conclude that for all $a \in F_\mathcal{P}$ and all $\omega_i \in \Omega$ there is an $a' \in F_\mathcal{P}$ such that $(\omega_i^\mathsf{x}(a), \omega_i^\mathsf{y}(a)) = a'$ holds. Using Definition 4.18 we know that $a' = (x', f_\mathcal{P}(x'))$, hence $\omega_i^\mathsf{x}(a) = x'$ and $\omega_i^\mathsf{y}(a) = f_\mathcal{P}(x')$. Therefore, we can conclude that $f_\mathcal{P}(\omega_i^\mathsf{x}(a)) = \omega_i^\mathsf{y}(a)$ holds for all $a \in F_\mathcal{P}$ and all $\omega_i \in \Omega$ if $F_\mathcal{P} = A$. $\square$

The underlying intuition is that every possible image of every point of the graph must be a point of the graph itself. Note that this theorem does not restrict the class of logic programs, i.e. it can be used to proof the equality of the attractors of iterated function systems and graphs of logic programs in general. The Equivalence Theorem will be used to prove the correctness of the approach presented in the following section.

The following theorem establishes a strong connection between logic programs and iterated function systems. Even though it does not define a method to construct an IFS, it explains why the plotted graphs of the programs are self-similar.

Theorem 5.3 *Let $\mathcal{P}$ be a logic program and $T_\mathcal{P}$ be the associated immediate consequence operator. Let $T_\mathcal{P}$ be continuous on $(I_\mathcal{P}, d_{\mathcal{P}B})$ where $d_{\mathcal{P}B}$ is constructed using an injective level mapping. Let $f_\mathcal{P}$ be the embedded $T_\mathcal{P}$-operator using base B. Then there is an IFS $\{(\mathbb{R}^2, d_2), \Omega\}$ with attractor A such that the graph of $f_\mathcal{P}$ coincides with A.*

Proof We take $\Omega = \{(\omega_i^\mathsf{x}, \omega_i^\mathsf{y})\}$ with $\omega_i^\mathsf{x} : \mathbb{R} \to \mathbb{R}$ being contractions with contractivity factor $\frac{1}{B^P}$, where $P \in \mathbb{N}$ is a free parameter. The set Ω^x as constructed in Definition 5.7 consists of such mappings and it can be shown that $\pi_\mathsf{x}(A) = D_\mathsf{f}$ (see Lemma 5.15 for details). Furthermore, we define ω^y as $\omega^\mathsf{y}(x) = f_\mathcal{P}(\omega^\mathsf{x}(x))$. In order to show that the IFS is hyperbolic it suffices to show that ω^y is a contraction, since ω^x is contractive by assumption.
From $T_\mathcal{P}$ being continuous on $(I_\mathcal{P}, d_{\mathcal{P}B})$ we get the continuity of $f_\mathcal{P}$ as shown in [Hit01, Chapter 9]. Since D_f is compact $f_\mathcal{P}$ must be Lipschitz-continuous, i.e. there is a constant L such that for all $x, y \in D_\mathsf{f}$ we have $d_1(f_\mathcal{P}(x), f_\mathcal{P}(y)) \leq L \cdot d_1(x, y)$. Using the definition of ω^y we get $d_1(\omega^\mathsf{y}(x_1), \omega^\mathsf{y}(x_2)) \leq L \cdot \frac{1}{B^P} \cdot d_1(x_1, x_2)$. Now, we can choose a P such that $L \cdot \frac{1}{B^P} < 1$ which turns ω^y into a contraction. Therefore, the constructed IFS is hyperbolic which allows the application of the Equivalence Theorem and it follows that the attractor of the IFS coincides with the graph of $f_\mathcal{P}$. □

Corollary 5.4 *For programs without local variables (including the acyclic ones) and programs for which ground($\mathcal{P}$) is finite, there exists an IFS with attractor A, such that the graph of the program coincides with A.*

Proof In [Sed95] it was shown that for all these programs the associated $T_\mathcal{P}$-operator is continuous, hence Theorem 5.3 and 5.2 are applicable. □

5.2. The Linear Case

The following section describes linear $IFS_{\mathcal{P}}$s in more detail. First, a simple example is given for motivation. Afterwards, the construction of a linear $IFS_{\mathcal{P}}$ is formalised and some properties are discussed. Finally, the Equivalence Theorem is used to prove the equality of the attractor and the graph of certain logic programs.

Example 5.5 The following example shows a logic program, the associated graph and an IFS for which the attractor is equal to the graph. The base is fixed with $B = 4$.

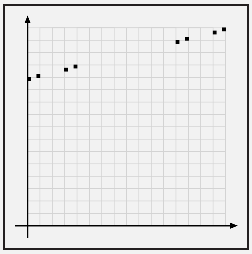
- Consider the following program $\mathcal{P}$ and the level mapping $|\cdot|$:

$\mathcal{P}$:

```
n(0).
n(s(X)) :- n(X).
```

$|\cdot|$: $|n(s^x(0))| = x + 1$

graph $F_{\mathcal{P}}$:



- The associated $T_{\mathcal{P}}$-operator can be written as:

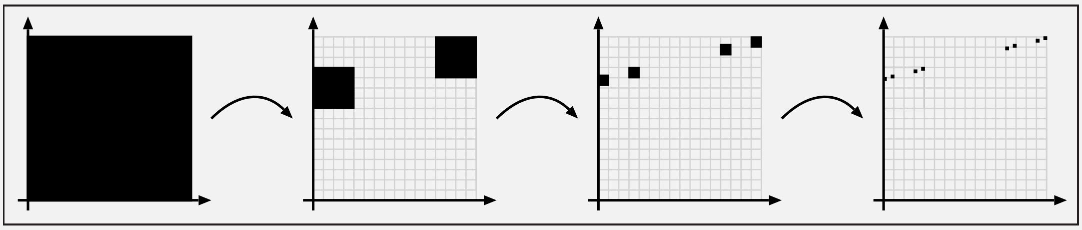
$$T_{\mathcal{P}} : I \mapsto \overset{I}{\underset{1}{\to}} \cup T_{\mathcal{P}}(\emptyset)$$

Using the embedding R and the equation $\mathbf{R}_{\cup}$ we derive $f_{\mathcal{P}}$:

$$f_{\mathcal{P}} : x \mapsto \frac{x}{B^1} + f_{\mathcal{P}}(0) \text{ and } f_{\mathcal{P}}(0) = \frac{1}{B} \qquad \text{because } \overset{I}{\underset{1}{\to}} \cap T_{\mathcal{P}}(\emptyset) = \emptyset.$$

- This function can be modelled using an IFS $= \{(\mathbb{R}^2, d_2), \Omega\}$ (the construction is shown in Example 5.11) with

$$\Omega = \left\{ \begin{pmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{4} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{3}{16} \end{pmatrix}, \begin{pmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{4} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{1}{4} \\ \frac{1}{4} \end{pmatrix} \right\}$$



This example shows that there is some evidence for the assumption that the graph of the logic program and the attractor of an appropriate constructed IFS coincide. At least they look very similar. In the sequel we will formalise the construction and prove the correctness.

5.2.1. The Construction of $IFS^l_{\mathcal{P}}$

This section describes a method to construct an $IFS_{\mathcal{P}}$ for a given program $\mathcal{P}$. The properties of the result are discussed within the next section. To formalise this construction some definitions are given and the algorithm used to construct the IFS from the example above is presented. Finally a construction is shown in detail.

Definition 5.6 ($IFS^l_{\mathcal{P}}$) *An $IFS_{\mathcal{P}}$ as introduced in Definition 5.1 is called a* linear $IFS_{\mathcal{P}}$ *(denoted $IFS^l_{\mathcal{P}}$) if the ω_i are defined as $\omega_i : \mathbb{R}^2 \to \mathbb{R}^2 : (x, y) \mapsto (\omega^{\mathsf{x}}_i(x), \omega^{\mathsf{y}}_i(y))$ with ω^{x}_i and ω^{y}_i defined as*

$$\omega^{\mathsf{x}}_i : \mathbb{R} \to \mathbb{R} : \frac{1}{B^P}x + d^{\mathsf{x}}_i \qquad\qquad \omega^{\mathsf{y}}_i : \mathbb{R} \to \mathbb{R} : \frac{1}{B^P}y + d^{\mathsf{y}}_i$$

where B denotes the base of the mapping R as introduced in Definition 4.1 and $d^{\mathsf{x}}_i, d^{\mathsf{y}}_i \in \mathbb{R}$.

In the definition above ω^{x}_i and ω^{y}_i are introduced as mappings from $\mathbb{R}$ to $\mathbb{R}$, instead of mapping $\mathbb{R}^2$ to $\mathbb{R}$ as required by Definition 5.1. This re-definition is made for simplicity and readability reasons and the mappings could easily be extended to fit the original definition.

The following definition constructs a set of mappings which will be used as ω^{x}_is defined above. These mappings are defined very similar to the production rules of the Cantor set as introduced in Section 3.3, since we want to create an IFS "built on top of" D_{f}.

Definition 5.7 (Ω^{x}_n) *Let R be the mapping as defined in Definition 4.1, using the base B. Then Ω^{x}_1 is defined as a set of mappings as follows:*

$$\Omega^{\mathsf{x}}_1 = \left\{ x \mapsto \frac{1}{B}x + 0; x \mapsto \frac{1}{B}x + \frac{1}{B} \right\}$$

For all $n > 1$ Ω^{x}_n is recursively defined as:

$$\begin{aligned}\Omega^{\mathsf{x}}_n &= \left\{ f \circ g \mid f \in \Omega^{\mathsf{x}}_1 \text{ and } g \in \Omega^{\mathsf{x}}_{n-1} \right\} \\ &= \left\{ x \mapsto \frac{1}{B}\omega(x) + 0 \mid \omega \in \Omega^{\mathsf{x}}_{n-1} \right\} \cup \left\{ x \mapsto \frac{1}{B}\omega(x) + \frac{1}{B} \mid \omega \in \Omega^{\mathsf{x}}_{n-1} \right\}\end{aligned}$$

Remark 5.8 *Note that every mapping ω^{x}_i is of the form $\omega^{\mathsf{x}}_i = \frac{1}{B^P}x + d^{\mathsf{x}}_i$, where d^{x}_i depends on the application of either the first or second mapping from Ω^{x}_1 during the construction. I.e. d^{x}_i can be written as $\sum_{j=1}^{P} a_j \cdot B^{-j}$, where $a_j \in \{0, 1\}$. Therefore, for each d^{x}_i there exists an interpretation $I_i = \{A \mid |A| = j, a_j = 1 \text{ for } 1 \le j \le P\}$ with $R(I_i) = d^{\mathsf{x}}_i$.*

The mappings in y-direction are constructed depending on the offsets of the mappings from Ω^{x}. These ω^{x}-mappings push the problem into "later" regions in the Herbrand base by performing right-shift operations, and add something to the front (the offsets d^{x}_i). Therefore, the mappings in y-direction should perform similar transformations.

Definition 5.9 (ω^{y}) *Let $\Omega^{\mathsf{x}}_n = \{\omega^{\mathsf{x}}_i\}$ be a set of mappings as constructed in Definition 5.7 using the mapping R with base B. Let $f_{\mathcal{P}}$ be the embedded version of the $T_{\mathcal{P}}$-operator associated with a logic program $\mathcal{P}$. Then ω^{y}_i is constructed from $\omega^{\mathsf{x}}_i = \frac{1}{B^P}x + d^{\mathsf{x}}_i$ as follows*

$$\omega^{\mathsf{y}}_i(y) = \frac{1}{B^P}y + f_{\mathcal{P}}(d^{\mathsf{x}}_i) - \frac{f_{\mathcal{P}}(0)}{B^P}.$$

This way of constructing the mappings allows to handle specific logic program, which are described later. The following algorithm describes the construction of an iterated function system for a given program. There is one free parameter not introduced yet, namely P. Its value depends on various conditions, which will be given in the sequel and summarised in Section 5.2.4.

Algorithm 5.10 (Construction of $IFS^l_{\mathcal{P}}$)
Let $\mathcal{P}$ be a logic program and $f_{\mathcal{P}}$ be its embedded $T_{\mathcal{P}}$-operator.

1. Choose a natural number (the *periodicity*) $P > 0$.
2. Compute Ω^{x}_P as explained in Definition 5.7.
3. Construct for each $\omega^{\mathsf{x}}_i \in \Omega^{\mathsf{x}}_P$ the corresponding $\omega^{\mathsf{y}}_i = \frac{1}{B^P}y + f_{\mathcal{P}}(d^{\mathsf{x}}_i) - \frac{f_{\mathcal{P}}(0)}{B^P}$ as in Definition 5.9 and define $\omega_i = (\omega^{\mathsf{x}}_i, \omega^{\mathsf{y}}_i)$.
4. Return the set $\Omega = \{\omega_i\}$ as mappings for the $IFS^l_{\mathcal{P}} = \{(\mathbb{R}^2, d_2), \Omega\}$.

Example 5.11 (5.5 continued – Construction of the $IFS^l_{\mathcal{P}}$)
Let $\mathcal{P}$ be the logic program and $|\cdot|$ be the level mapping from Example 5.5.

1. The periodicity chosen is $P = 1$.
2. And therefore, the set of mappings in x-direction is defined as:

$$\Omega^{\mathsf{x}}_P = \left\{ \omega^{\mathsf{x}}_1(x) = \frac{1}{B}x + 0; \quad \omega^{\mathsf{x}}_2(x) = \frac{1}{B}x + \frac{1}{B} \right\}$$

3. The corresponding mappings in y-direction are:

$$\begin{aligned} \omega^{\mathsf{y}}_1(y) &= \frac{1}{B}y + f_{\mathcal{P}}(0) - \frac{f_{\mathcal{P}}(0)}{B} \\ &= \frac{1}{B}y + \frac{1}{B} - \frac{\frac{1}{B}}{B} \end{aligned} \qquad \begin{aligned} \omega^{\mathsf{y}}_1(y) &= \frac{1}{B}y + f_{\mathcal{P}}\left(\frac{1}{B}\right) - \frac{f_{\mathcal{P}}(0)}{B} \\ &= \frac{1}{B}y + \frac{\frac{1}{B}}{B} + \frac{1}{B} - \frac{\frac{1}{B}}{B} \end{aligned}$$

4. For $B = 4$ we get $IFS_{\mathcal{P}} = \{(\mathbb{R}^2, d_2), \Omega)\}$ with

$$\Omega = \left\{ \begin{pmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{4} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{3}{16} \end{pmatrix}, \begin{pmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{4} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{1}{4} \\ \frac{1}{4} \end{pmatrix} \right\}$$

5.2.2. Properties of $IFS^{l}_{\mathcal{P}}$

This section is concerned with the properties of the $IFS_{\mathcal{P}}$s constructed above. First, it is shown that the result of Algorithm 5.10 actually is a hyperbolic IFS. Afterwards, some propositions and lemmas are given which are used for proofs later in this thesis.

Theorem 5.12 *Let $\{(\mathbb{R}^2, d_2), \Omega\}$ be an $IFS_{\mathcal{P}}$ constructed as given in Algorithm 5.10, with $\Omega = \{(\omega_i^{\mathsf{x}}, \omega_i^{\mathsf{y}})\}$. Let d_1 and d_2 denote the 1- and 2-dimensional Euclidean metric respectively. Then ω_i^{x}, ω_i^{y} and $\omega_i = (\omega_i^{\mathsf{x}}, \omega_i^{\mathsf{y}})$ are contractions on $(\mathbb{R}, d_1)$ and $(\mathbb{R}^2, d_2)$, and $\{(\mathbb{R}^2, d_2), \Omega\}$ is a (hyperbolic) iterated function system as introduced in Definition 3.31.*

Proof It is easy to see that ω_i^{x} and ω_i^{y} are contractions on $(\mathbb{R}, d_1)$, as for their (contractivity) factors $0 < s = \frac{1}{B^P} < 1$ will hold for all $B > 1$ and $P > 0$. To show that ω_i is a contraction on $(\mathbb{R}^2, d_2)$ we need to show that for all ω_i and all $(x_1, y_1), (x_2, y_2) \in \mathbb{R}^2$ $d_2(\omega_i(x_1, y_1), \omega_i(x_2, y_2)) \leq s \cdot d_2((x_1, y_1), (x_2, y_2))$ holds. Using the definition of d_2 and ω_i, we get $d_2(\omega_i(x_1, y_1), \omega_i(x_2, y_2)) = \sqrt{(\omega_i^{\mathsf{x}}(x_1) - (\omega_i^{\mathsf{x}}(x_2))^2 + (\omega_i^{\mathsf{y}}(y_1) - (\omega_i^{\mathsf{y}}(y_2))^2}$. And knowing that ω_i^{x} and ω_i^{y} are contractions with contractivity factor $\frac{1}{B^P}$ allows to conclude that $d_2(\omega_i(x_1, y_1), \omega_i(x_2, y_2)) \leq \frac{1}{B^P} \cdot \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ which is equal to $\frac{1}{B^P} \cdot d_2((x_1, y_1), (x_2, y_2))$. Therefore, we can conclude that all ω_i are contractions. Taking into account that $(\mathbb{R}^2, d_2)$ is a complete metric space we can conclude that $\{(\mathbb{R}^2, d_2), \Omega\}$ is a (hyperbolic) iterated function system. □

This theorem tells us that the $IFS_{\mathcal{P}}$s actually are hyperbolic iterated function systems, i.e. a set of contractive mappings defined on a complete metric space. Therefore, we know that there is a uniquely determined compact subset of $\mathbb{R}^2$, namely the attractor A, to which for any $B \in \mathcal{H}(\mathbf{X})$ the sequence $\Omega(B), \Omega(\Omega(B)), \ldots$ converges with respect to h_{d_2}.

Proposition 5.13 *Let R be the mapping as defined in Definition 4.1, let $\dot{\rightarrow}$ be defined as in Section 4.1.3 and let d_i^{x} be constructed using P as given by Algorithm 5.10. Then the following equation holds for all $I \in I_{\mathcal{P}}$:*

$$\overset{I}{\underset{P}{\rightarrow}} \cap R^{-1}(d_i^{\mathsf{x}}) = \emptyset \tag{sep}$$

Proof From Remark 5.8 we know that $d_i^{\mathsf{x}} = \sum_{j=1}^{P} a_j \cdot B^{-j}$ where $a_j \in \{0, 1\}$ and that there is an interpretation $I_i = \{A \mid |A| = j \text{ and } a_j = 1\}$ such that $R(I_i) = d_i^{\mathsf{x}}$. Using Proposition 4.14 we know, that all the atoms contained in $\overset{I}{\underset{P}{\rightarrow}}$ must have a level larger than P and, therefore, $\overset{I}{\underset{P}{\rightarrow}}$ does not contain any atom A with $1 \leq A \leq P$. Therefore, the intersection of $\overset{I}{\underset{P}{\rightarrow}}$ and $J = R^{-1}(d_i^{\mathsf{x}})$ must be empty. □

Proposition 5.14 *Let ω_i^{x} and ω_i^{y} be defined as above. Then both mappings are injective.*

Proof This holds since both mappings are linear. □

Lemma 5.15 *Let $\Omega_P^{\mathsf{x}} = \{\omega_i^{\mathsf{x}}\}$ be defined as above and $D_{\mathsf{f}} \in \mathbb{R}$ be the range of the mapping R as introduced in Definition 4.3. Then D_{f} is the attractor of the IFS $\{(\mathbb{R}, d), \Omega_P^{\mathsf{x}}\}$.*

Proof From Theorem 5.12 we know that all ω_i^{x} are contractive mappings on the complete metric space $(\mathbb{R}, d_1)$. Therefore, $\{(\mathbb{R}, d_1), \Omega_P^{\mathsf{x}}\}$ is a hyperbolic IFS according to Definition 3.31, and we can conclude that the attractor is uniquely determined. We simply need to prove that D_{f} is a fixed point of Ω_P^{x}, i.e. $D_{\mathsf{f}} = \Omega_P^{\mathsf{x}}(D_{\mathsf{f}})$. This can be done by showing that **(i)** $D_{\mathsf{f}} \subseteq \Omega_P^{\mathsf{x}}(D_{\mathsf{f}})$ and **(ii)** $D_{\mathsf{f}} \supseteq \Omega_P^{\mathsf{x}}(D_{\mathsf{f}})$.

(i) To prove that $D_{\mathsf{f}} \subseteq \Omega_P^{\mathsf{x}}(D_{\mathsf{f}})$ we need to show that for all $x \in D_{\mathsf{f}}$ there exists an $\omega_i^{\mathsf{x}} \in \Omega_P^{\mathsf{x}}$ and an $x' \in D_{\mathsf{f}}$, such that $x = \omega_i^{\mathsf{x}}(x') = \frac{1}{B^P}x' + d_i^{\mathsf{x}}$, which is equivalent to $x' = (x - d_i^{\mathsf{x}}) \cdot B^P$. Let ω_i^{x} be the mapping for which $R^{-1}(d_i^{\mathsf{x}})$ and $R^{-1}(x)$ agree on all atoms A with $|A| \leq P$. This mapping exists, as for any possible combination of atoms with level $\leq P$ there is an adequate mapping ω_i^{x} and, therefore, $x - d_i^{\mathsf{x}} \in D_{\mathsf{f}}$. Since the interpretation $R^{-1}(x - d_i^{\mathsf{x}})$ does not contain any atom with level $\leq P$, it can be shifted P times to the left analogous to Definition 4.13. This corresponds to a multiplication with B^P, hence $x' = (x - d_i^{\mathsf{x}}) \cdot B^P$ is an element of D_{f} for all $x \in D_{\mathsf{f}}$ and $D_{\mathsf{f}} \subseteq \Omega_P^{\mathsf{x}}(D_{\mathsf{f}})$.

(ii) To show that $D_{\mathsf{f}} \supseteq \Omega_P^{\mathsf{x}}(D_{\mathsf{f}})$ we need to show that for all $\omega_i^{\mathsf{x}} \in \Omega^{\mathsf{x}}$ and for all $x \in D_{\mathsf{f}}$ there exists an $x' \in D_{\mathsf{f}}$, such that $x' = \omega_i^{\mathsf{x}}(x) = \frac{1}{B^P}x + d_i^{\mathsf{x}}$. Applying R^{-1} we get $R^{-1}(x') = R^{-1}(\frac{1}{B^P}x + d_i^{\mathsf{x}})$. From Proposition 5.13 and Definition 4.13 we know that $R^{-1}(\frac{1}{B^P}x) \cap R^{-1}(d_i^{\mathsf{x}}) = \emptyset$. Furthermore, we can conclude that $R^{-1}(x') = R^{-1}(\frac{1}{B^P}x + d_i^{\mathsf{x}}) = R^{-1}(\frac{1}{B^P}x) \cup d_i^{\mathsf{x}}$ (see $\mathbf{R}_{\cup}$ in Proposition 4.12) and hence $x' \in D_{\mathsf{f}}$. Therefore, $D_{\mathsf{f}} \supseteq \Omega_P^{\mathsf{x}}(D_{\mathsf{f}})$ holds, which ends the proof. $\square$

Corollary 5.16 *Let $\pi_{\mathsf{x}}(\cdot)$ denote the projection onto the x-axis and A be the attractor of an $IFS_{\mathcal{P}}$ as constructed above. Then $\pi_{\mathsf{x}}(A) = D_{\mathsf{f}}$.*

Lemma 5.17 *Let $\Omega_P^{\mathsf{x}} = \{\omega_i^{\mathsf{x}}\}$ be a set of mappings as introduced in Definition 5.7 and $D_{\mathsf{f}B} \subset \mathbb{R}$ be the range of the mapping R for base B. Then the different images of $D_{\mathsf{f}B}$ do not intersect, i.e. for all $0 < i \neq j \leq n$ we have $\omega_i^{\mathsf{x}}(D_{\mathsf{f}B}) \cap \omega_j^{\mathsf{x}}(D_{\mathsf{f}B}) = \emptyset$.*

Proof Assume that the images overlap, i.e. that there are $i \neq j$ such that $\omega_i^{\mathsf{x}}(D_{\mathsf{f}B}) \cap \omega_j^{\mathsf{x}}(D_{\mathsf{f}B}) \neq \emptyset$. Then there must exist $x, y \in D_{\mathsf{f}B}$ such that $\omega_i^{\mathsf{x}}(x) = \omega_j^{\mathsf{x}}(y)$. This is equal to $\frac{x}{B^P} + d_i^{\mathsf{x}} = \frac{y}{B^P} + d_j^{\mathsf{x}}$ and since $x, y \in D_{\mathsf{f}B}$ there are $I = R^{-1}(x)$ and $J = R^{-1}(y) \in I_{\mathcal{P}}$ such that $R\left(\frac{I}{P}\right) + d_i^{\mathsf{x}} = R\left(\frac{J}{P}\right) + d_j^{\mathsf{x}}$ holds. Using the observation from Remark 5.8 we can conclude that there must be $D_i, D_j \in I_{\mathcal{P}}$ such that $R(D_i) = d_i^{\mathsf{x}}$ and $R(D_j) = d_j^{\mathsf{x}}$. Furthermore, (using $\mathbf{R}_{\cup}$ and Proposition 5.13) we get $R\left(\frac{I}{P} \cup D_i\right) = R\left(\frac{J}{P} \cup D_j\right)$. But this can never be true, as we know that $D_i \neq D_j$. Therefore, there must be at least one atom on which $\frac{I}{P} \cup D_i$ and $\frac{J}{P} \cup D_j$ differ. Hence, the assumption made in the beginning of the proof is wrong and the intersection of $\omega_i^{\mathsf{x}}(D_{\mathsf{f}B})$ and $\omega_j^{\mathsf{x}}(D_{\mathsf{f}B})$ must be empty. $\square$

Corollary 5.18 *Let $\{(\mathbb{R}^2, d_2), \Omega\}$ be an $IFS_{\mathcal{P}}$, as constructed in Algorithm 5.10. Then $\{(\mathbb{R}, d_1), \Omega^{\mathsf{x}}\}$ and $\{(\mathbb{R}^2, d_2), \Omega\}$ are totally disconnected as introduced in Definition 3.38.*

Since the iterated function systems constructed using Algorithm 5.10 are totally disconnected we can apply Theorem 3.43 to compute their fractal dimension D. Using base B and periodicity P the algorithm creates 2^P mappings, which are similitudes with scaling factor $\frac{1}{B^P}$, hence $2^P \cdot \left(\frac{1}{B^P}\right)^D = 1$. Therefore, we can conclude that $D = \frac{\ln(2^P)}{\ln(B^P)} = \frac{\ln 2}{\ln B}$, which coincides with the fractal dimension of the Cantor set $\mathcal{C}$, constructed by repeatedly removing $\frac{B-2}{B} \cdot 100\%$ from the middle.

5.2.3. Using the Equivalence Theorem to Prove Correctness

Within this section the Equivalence Theorem is applied to show that Algorithm 5.10 can be used to construct an IFS for a given program, such that attractor and graph coincide. The following two definitions are used to characterise a certain class of logic programs. They arose from the proof of Theorem 5.21. We cannot give an intuition yet, but as we will see in that proof they define sufficient conditions under which the required equation $f_\mathcal{P}(\omega_i^{\mathsf{x}}(x)) = \omega_i^{\mathsf{y}}(f_\mathcal{P}(x))$ holds.

Definition 5.19 (separability) *Let $\mathcal{P}$ be a logic program, P be a natural number and J be an interpretation. Then the program $\mathcal{P}$ is said to be* separable *with respect to P and J, if the following conditions hold for all $I \in I_\mathcal{P}$:*

$$T_\mathcal{P}\left(\xrightarrow[P]{I}\right) \cap T_\mathcal{P}(J) = T_\mathcal{P}(\emptyset) \qquad (\mathbf{sep_1})$$

$$T_\mathcal{P}\left(\xrightarrow[P]{I} \cup J\right) = T_\mathcal{P}\left(\xrightarrow[P]{I}\right) \cup T_\mathcal{P}(J) \qquad (\mathbf{sep_2})$$

$$\xrightarrow[P]{I} \cap T_\mathcal{P}(\emptyset) = \emptyset \qquad (\mathbf{sep_3})$$

Definition 5.20 (linearity) *Let $\mathcal{P}$ be a logic program and $P > 0$ be a natural number. Then the program $\mathcal{P}$ is said to be* linear *with respect to P, if the following condition holds for all $I \in I_\mathcal{P}$:*

$$T_\mathcal{P}\left(\xrightarrow[P]{I}\right) = \left(\xrightarrow[P]{T_\mathcal{P}(I)} \setminus \xrightarrow[P]{T_\mathcal{P}(\emptyset)}\right) \cup T_\mathcal{P}(\emptyset) \qquad (\mathbf{lin})$$

The conditions introduced in the definitions above are very restrictive and it is not clear yet which class of logic programs is defined by them. In Example 5.22 it is shown that the program from Example 5.5 is separable and linear, i.e. the set of linear and separable logic program is not empty. For these programs the following theorem shows that the attractor of the IFS constructed using Algorithm 5.10 coincides with their graph.

Theorem 5.21 *Let $\mathcal{P}$ be a logic program, $F_{\mathcal{P}}$ be the graph of its embedded $T_{\mathcal{P}}$-operator $f_{\mathcal{P}}$ and $\{(\mathbb{R}^2, d_2), \Omega\}$ be an iterated function system as constructed in Algorithm 5.10, using the periodicity P. Furthermore, let $\mathcal{P}$ be separable with respect to P and $R^{-1}(d_i^{\mathsf{x}})$, for all d_i^{x} in Ω, and let $\mathcal{P}$ be linear with respect to P. Then $F_{\mathcal{P}}$ is the attractor of $\{(\mathbb{R}^2, d_2), \Omega\}$.*

Proof From the Equivalence Theorem (5.2) follows that $F_{\mathcal{P}}$ is the attractor of $IFS_{\mathcal{P}}$ if and only if for all $x \in D_{\mathsf{f}}$ and all $\omega_i \in \Omega$ the equation $f_{\mathcal{P}}(\omega_i^{\mathsf{x}}(x)) = \omega_i^{\mathsf{y}}(f_{\mathcal{P}}(x))$ holds, which will shown now.

$$f_{\mathcal{P}}(\omega_i^{\mathsf{x}}(x)) = f_{\mathcal{P}}\left(\frac{1}{B^P}x + d_i^{\mathsf{x}}\right) = R\left(T_{\mathcal{P}}\left(R^{-1}\left(\frac{1}{B^P}x + d_i^{\mathsf{x}}\right)\right)\right)$$

using $\mathbf{R}_{+}$ and **sep** (from Proposition 5.13) this can be transformed into

$$= R\left(T_{\mathcal{P}}\left(R^{-1}\left(\frac{1}{B^P}x\right) \cup R^{-1}(d_i^{\mathsf{x}})\right)\right)$$

Definition 4.13 allows us to conclude that

$$= R\left(T_{\mathcal{P}}\left(\xrightarrow[P]{R^{-1}(x)} \cup R^{-1}(d_i^{\mathsf{x}})\right)\right) \overset{\mathbf{sep_2}}{=} R\left(T_{\mathcal{P}}\left(\xrightarrow[P]{R^{-1}(x)}\right) \cup T_{\mathcal{P}}(R^{-1}(d_i^{\mathsf{x}}))\right)$$

knowing that $\mathbf{R}_{\cup}$ and $\mathbf{sep_1}$ hold, we can conclude

$$= R\left(T_{\mathcal{P}}\left(\xrightarrow[P]{R^{-1}(x)}\right)\right) + R(T_{\mathcal{P}}(R^{-1}(d_i^{\mathsf{x}}))) - R(T_{\mathcal{P}}(\emptyset))$$

lin gives us the following

$$= R\left(\left(\xrightarrow[P]{T_{\mathcal{P}}(R^{-1}(x))} \setminus \xrightarrow[P]{T_{\mathcal{P}}(\emptyset)}\right) \cup T_{\mathcal{P}}(\emptyset)\right) + R(T_{\mathcal{P}}(R^{-1}(d_i^{\mathsf{x}}))) - R(T_{\mathcal{P}}(\emptyset))$$

using $\mathbf{R}_{\cup}$, $\mathbf{sep_3}$ and Definition 4.16 gives us

$$= R\left(\xrightarrow[P]{T_{\mathcal{P}}(R^{-1}(x))} \setminus \xrightarrow[P]{T_{\mathcal{P}}(\emptyset)}\right) + R(T_{\mathcal{P}}(\emptyset)) + f_{\mathcal{P}}(d_i^{\mathsf{x}}) - R(T_{\mathcal{P}}(\emptyset))$$

using $\mathbf{R}_{\setminus}$ we can conclude

$$= R\left(\xrightarrow[P]{T_{\mathcal{P}}(R^{-1}(x))}\right) - R\left(\xrightarrow[P]{T_{\mathcal{P}}(\emptyset)}\right) + R(T_{\mathcal{P}}(\emptyset)) + f_{\mathcal{P}}(d_i^{\mathsf{x}}) - R(T_{\mathcal{P}}(\emptyset))$$

from Definition 4.13 we get

$$= \frac{1}{B^P} f_{\mathcal{P}}(x) + f_{\mathcal{P}}(d_i^{\mathsf{x}}) - \frac{f_{\mathcal{P}}(0)}{B^P}$$

using Definition 4.16 we get

$$= \omega_i^{\mathsf{y}}(f_{\mathcal{P}}(x))$$

$\square$

Example 5.22 (5.11 continued) Here it is shown that the program from Example 5.5 is linear with respect to $P = 1$ and separable with respect to $I_1 = R^{-1}(d_1^{\mathsf{x}}) = \emptyset$ and $I_2 = R^{-1}(d_2^{\mathsf{x}}) = \{n(0)\}$. The parts of the interpretations we are not primarily concerned with are denoted ... and for those it is obvious that they are equal within each formula.

	$I_1 = \emptyset$	$I_2 = \{n(0)\}$
sep₁	$T_{\mathcal{P}}\left(\xrightarrow[1]{I_1}\right) \cap T_{\mathcal{P}}(\emptyset) = T_{\mathcal{P}}(\emptyset)$ $\{n(0), \cancel{n(s(0))}, \dots\} \cap \{n(0)\} = \{n(0)\}$	$T_{\mathcal{P}}\left(\xrightarrow[1]{I_2}\right) \cap T_{\mathcal{P}}(\{n(0)\}) = T_{\mathcal{P}}(\emptyset)$ $\{n(0), \cancel{n(s(0))}, \dots\} \cap \{n(0), n(s(0))\} = \{n(0)\}$
sep₂	$T_{\mathcal{P}}\left(\xrightarrow[1]{I_1} \cup \emptyset\right) = T_{\mathcal{P}}\left(\xrightarrow[1]{I_1}\right) \cup T_{\mathcal{P}}(\emptyset)$ $\{n(0), \dots\} = \{n(0), \dots\} \cup \{n(0)\}$	$T_{\mathcal{P}}\left(\xrightarrow[1]{I_2} \cup \{n(0)\}\right) = T_{\mathcal{P}}\left(\xrightarrow[1]{I_2}\right) \cup T_{\mathcal{P}}(\{n(0)\})$ $\{n(0), n(s(0)), \dots\} = \{n(0), \cancel{n(s(0))}, \dots\} \cup \{n(0), n(s(0))\}$

	for all $I \in I_{\mathcal{P}}$
sep₃	$\xrightarrow[1]{I} \cap T_{\mathcal{P}}(\emptyset) = \emptyset$ $\{\cancel{n(0)}, \dots\} \cap \{n(0)\} = \emptyset$
lin	$T_{\mathcal{P}}\left(\xrightarrow[1]{I}\right) = \left(\xrightarrow[1]{T_{\mathcal{P}}(I)} \setminus \xrightarrow[1]{T_{\mathcal{P}}(\emptyset)}\right) \cup T_{\mathcal{P}}(\emptyset)$ $\{n(0), \cancel{n(s(0))}, \dots\} = \{\cancel{n(0)}, n(s(0)), \dots\} \setminus \{n(s(0))\} \cup \{n(0)\}$

Hence the program is separable and linear with respect to P, $I_1 = R^{-1}(d_1^{\mathsf{x}})$ and $I_2 = R^{-1}(d_2^{\mathsf{x}})$. Therefore, we can conclude that the graph of the program and the attractor of the IFS coincide as shown in Theorem 5.21.

5.2.4. Summary and Conclusion

Note that Algorithm 5.10 produces an $IFS_{\mathcal{P}}^{l}$ for any periodicity P. Unfortunately, there is no way known yet to compute this periodicity such that the attractor and the graph of the program coincide. So far this coincidence can only be assured, if the program is separable and linear with respect to P. That is, we can check whether a certain periodicity is "valid" for a given program, but cannot compute the set of the "valid" ones. It turned out, that for a level mapping of the form $|A| = p \cdot x + k$, with x being a parameter and k being a constant, p is a good guess. Example 5.5 and 5.23 show instances of such level mappings.

Within this section it was shown, that the Equivalence Theorem can be applied to prove the coincidence of the graph of a logic program and the attractor of an appropriately constructed IFS. For that purpose the programs have to meet two conditions, namely they must be linear and separable. This conditions are very strong but the class of fulfilling programs is not empty.

Furthermore, this section gives an idea how to proceed in the future. In order to broaden the class of programs, suitable conditions must be defined such that the equivalence of $f_{\mathcal{P}}(\omega_i^{\mathsf{x}}(a))$ and $\omega_i^{\mathsf{y}}(a)$ can be shown. The following section shows an example which is neither linear nor separable, but for which the equivalence can be shown nevertheless.

5.3. A Nonlinear Example

The previous section was concerned with linear programs only. For those programs which are separable according to Definition 5.19 and linear according to Definition 5.20 it was shown that the attractor of the corresponding $IFS_{\mathcal{P}}$ coincides with the graphs of their embedded $T_{\mathcal{P}}$-operators. But as we will see now the constraints made are too strong, since there are non-linear logic programs for which Algorithm 5.10 constructs a valid $IFS_{\mathcal{P}}$.

First, an example is given, which re-introduces the logic program from Example 2.12 and gives the $IFS_{\mathcal{P}}$ constructed according to Algorithm 5.10. Afterwards, it is shown that the program does not fulfil the conditions introduced in the last section. And finally, it is shown that the graph of the program is the attractor of the $IFS_{\mathcal{P}}$.

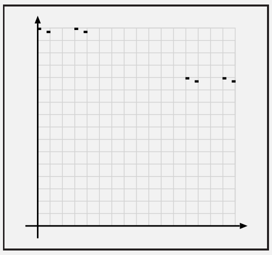
Example 5.23 The following example shows a logic program $\mathcal{P}$, the associated graph $F_{\mathcal{P}}$ and an IFS with an attractor A equal to $F_{\mathcal{P}}$. The base was fixed with $B = 4$.

- Consider the following program $\mathcal{P}$ and the level mapping $|\cdot|$:

$\mathcal{P}$:

```
e(0).
e(s(X)) :- not e(X).
o(X) :- not e(X).
```

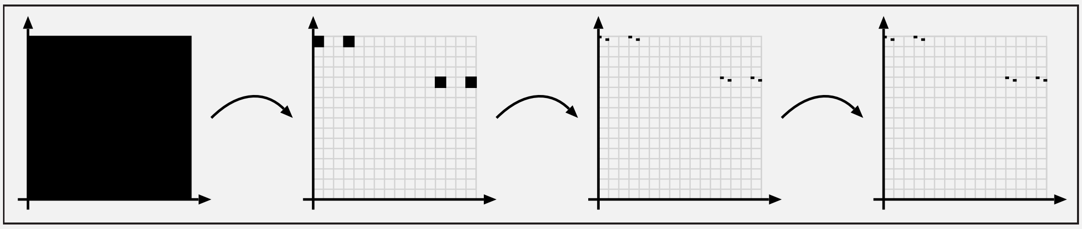
graph $F_{\mathcal{P}}$:



$|\cdot|$: $|e(s^x(0))| = 2 \cdot x + 1$
$|o(s^x(0))| = 2 \cdot x + 2$

- $IFS_{\mathcal{P}} = \{(\mathbb{R}^2, d_2), \Omega\}$ as constructed in Algorithm 5.10 with $P = 2$:

$$\Omega = \left\{ \begin{pmatrix} \frac{1}{16} & 0 \\ 0 & \frac{1}{16} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{5}{16} \end{pmatrix}, \begin{pmatrix} \frac{1}{16} & 0 \\ 0 & \frac{1}{16} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{1}{16} \\ \frac{5}{16} \end{pmatrix}, \right.$$
$$\left. \begin{pmatrix} \frac{1}{16} & 0 \\ 0 & \frac{1}{16} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{1}{4} \\ \frac{15}{64} \end{pmatrix}, \begin{pmatrix} \frac{1}{16} & 0 \\ 0 & \frac{1}{16} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{5}{16} \\ \frac{15}{64} \end{pmatrix} \right\}$$



It is easy to show that this program is not linear with respect to $P = 2$, as defined in Definition 5.20. To show this we take $I = \{e(0)\}$, which gives us $\frac{I}{2} = \{e(s(0))\}$, and consequently $T_{\mathcal{P}}(\emptyset) = \mathcal{B}_{\mathcal{P}}$ and $T_{\mathcal{P}}(\frac{I}{2}) = T_{\mathcal{P}}(\{e(s(0))\}) = \mathcal{B}_{\mathcal{P}} \setminus \{e(s(s(0))), o(s(0))\}$. Therefore, Equation **lin** does not hold for the program from the example above. Furthermore, the program is not separable with respect to $I = \{e(0)\}$ as introduced in Definition 5.19, since $\frac{I}{P} \cap T_{\mathcal{P}}(\emptyset) = \{e(s(0))\} \cap \mathcal{B}_{\mathcal{P}} \neq \emptyset$, which invalidates $\mathbf{sep_3}$.

But we can still show that the graph of the program is the attractor of the $IFS_{\mathcal{P}}$, which can be proven by showing that $f_{\mathcal{P}}(\omega_i^{\mathsf{x}}(x)) = \omega_i^{\mathsf{y}}(f_{\mathcal{P}}(y))$ holds for all x, y and

ω_i as stated in the Equivalence Theorem. Using the definition of $f_{\mathcal{P}}$ we know that $T_{\mathcal{P}}(I) = R^{-1}(f_{\mathcal{P}}(R(I)))$. A similar embedding can be used for ω_i^{x} and ω_i^{y}, namely $\mathsf{w}_i^{\mathsf{x}}(I) = R^{-1}(\omega_i^{\mathsf{x}}(R(I)))$ and $\mathsf{w}_i^{\mathsf{y}}(I) = R^{-1}(\omega_i^{\mathsf{y}}(R(I)))$ respectively. Therefore, the equation $f_{\mathcal{P}}(\omega_i^{\mathsf{x}}(x)) = \omega_i^{\mathsf{y}}(f_{\mathcal{P}}(y))$ can be proven by showing that $T_{\mathcal{P}}(\mathsf{w}_i^{\mathsf{x}}(I)) = \mathsf{w}_i^{\mathsf{y}}(T_{\mathcal{P}}(I))$ holds for all i and all $I \in I_{\mathcal{P}}$. These equations for the even and odd numbers are (informally) shown in Figure 11, where base $B = 4$ was used.

<table>
<tr><td>$\omega_1^{\mathsf{x}}(x) = \frac{x}{16} + 0$
$\mathsf{w}_1^{\mathsf{x}}(I) = \overset{I}{\underset{2}{\rightarrow}}$</td><td>$\omega_1^{\mathsf{y}}(x) = \frac{x}{16} + \frac{5}{16}$
$\mathsf{w}_1^{\mathsf{y}}(I) = \overset{I}{\underset{2}{\rightarrow}} \cup \{e(0), o(0)\}$</td></tr>
<tr><td colspan="2">$T_{\mathcal{P}}(\mathsf{w}_1^{x}(I)) = T_{\mathcal{P}}\left(\overset{I}{\underset{2}{\rightarrow}}\right) = \overset{T_{\mathcal{P}}(I)}{\underset{2}{\rightarrow}} \cup \{e(0), o(0)\} = \mathsf{w}_1^{\mathsf{y}}(T_{\mathcal{P}}(I))$</td></tr>
<tr><td>$\omega_2^{\mathsf{x}}(x) = \frac{x}{16} + \frac{1}{16}$
$\mathsf{w}_2^{\mathsf{x}}(I) = \overset{I}{\underset{2}{\rightarrow}} \cup \{o(0)\}$</td><td>$\omega_2^{\mathsf{y}}(x) = \frac{x}{16} + \frac{5}{16}$
$\mathsf{w}_2^{\mathsf{y}}(I) = \overset{I}{\underset{2}{\rightarrow}} \cup \{e(0), o(0)\}$</td></tr>
<tr><td colspan="2">$T_{\mathcal{P}}(\mathsf{w}_2^{x}(I)) = T_{\mathcal{P}}\left(\overset{I}{\underset{2}{\rightarrow}} \cup \{o(0)\}\right) = \overset{T_{\mathcal{P}}(I)}{\underset{2}{\rightarrow}} \cup \{e(0), o(0)\} = \mathsf{w}_2^{\mathsf{y}}(T_{\mathcal{P}}(I))$</td></tr>
<tr><td>$\omega_3^{\mathsf{x}}(x) = \frac{x}{16} + \frac{1}{4}$
$\mathsf{w}_3^{\mathsf{x}}(I) = \overset{I}{\underset{2}{\rightarrow}} \cup \{e(0)\}$</td><td>$\omega_3^{\mathsf{y}}(x) = \frac{x}{16} + \frac{15}{64}$
$\mathsf{w}_3^{\mathsf{y}}(I) = \overset{I}{\underset{2}{\rightarrow}} \setminus \{e(s(0))\} \cup \{e(0)\}$</td></tr>
<tr><td colspan="2">$T_{\mathcal{P}}(\mathsf{w}_3^{x}(I)) = T_{\mathcal{P}}\left(\overset{I}{\underset{2}{\rightarrow}} \cup \{e(0)\}\right) = \overset{T_{\mathcal{P}}(I)}{\underset{2}{\rightarrow}} \setminus \{e(s(0))\} \cup \{e(0)\} = \mathsf{w}_3^{\mathsf{y}}(T_{\mathcal{P}}(I))$</td></tr>
<tr><td>$\omega_4^{\mathsf{x}}(x) = \frac{x}{16} + \frac{5}{16}$
$\mathsf{w}_4^{\mathsf{x}}(I) = \overset{I}{\underset{2}{\rightarrow}} \cup \{e(0), o(0)\}$</td><td>$\omega_4^{\mathsf{y}}(x) = \frac{x}{16} + \frac{15}{64}$
$\mathsf{w}_4^{\mathsf{y}}(I) = \overset{I}{\underset{2}{\rightarrow}} \setminus \{e(s(0))\} \cup \{e(0)\}$</td></tr>
<tr><td colspan="2">$T_{\mathcal{P}}(\mathsf{w}_4^{x}(I)) = T_{\mathcal{P}}\left(\overset{I}{\underset{2}{\rightarrow}} \cup \{e(0), o(0)\}\right) = \overset{T_{\mathcal{P}}(I)}{\underset{2}{\rightarrow}} \setminus \{e(s(0))\} \cup \{e(0)\} = \mathsf{w}_4^{\mathsf{y}}(T_{\mathcal{P}}(I))$</td></tr>
</table>

Figure 11: $T_{\mathcal{P}}(\mathsf{w}_i^{\mathsf{x}}(I)) = \mathsf{w}_i^{\mathsf{y}}(T_{\mathcal{P}}(I))$ holds for the even and odd numbers

The example discussed in this section shows that the conditions imposed in Section 5.2 are too strong. They were defined in order to apply the Equivalence Theorem. But Algorithm 5.10 is valid for more programs than allowed by these conditions. Unfortunately, it is not clear yet for which class of logic programs the algorithm works correctly. This remains to be clarified in the future.

5.4. Fractal Interpolation for Arbitrary Programs

This section presents another approach to link logic programs and iterated function systems, namely a fractal interpolation similar to the technique introduced in Section 3.5. A method is presented which creates an iterated function system approximating the graph of a program to a certain degree. The construction of the IFS is described and some properties are discussed.

Algorithm 5.24 (Construction of Interpolation Data)
This algorithm computes a set of interpolation data for a given program $\mathcal{P}$, where R denotes the mapping from Definition 4.1.

1. Choose a natural number (the *accuracy*) $P > 0$.
2. Compute the set $D = \{A \mid |A| \leq P \text{ and } A \in \mathcal{B}_\mathcal{P}\}$ and its powerset $\mathcal{D} = \mathcal{P}(D)$.
3. For any set $X_i \in \mathcal{D}$ compute $Y_i = T_\mathcal{P}(X_i)$ with respect to the program $\mathcal{P}$.
4. Return the sequence $\{(R(X_i), R(Y_i))\}$, with $R(X_i) < R(X_j)$ for all $i < j$.

For the sequel we will denote the set of interpolation data produced by the algorithm above as $D = \{(d_i^x, d_i^y)\}$. This decision will be clear after the next definition, as there is a strong relation to the $IFS_\mathcal{P}$s constructed in Section 5.2.

Definition 5.25 (*$IIFS_\mathcal{P}$*) *Let $\{(d_i^x, d_i^y) \mid 1 \leq i \leq 2^P\}$ be a sequence of interpolation data constructed from the program $\mathcal{P}$ using accuracy P. Let $f_\mathcal{P}$ be the embedded $T_\mathcal{P}$-operator associated with the program $\mathcal{P}$ using the mapping R with base B. Then $\{(\mathbb{R}^2, d_2), \Omega\}$ is called an* interpolating iterated function system *($IIFS_\mathcal{P}$), with $\Omega = \{\omega_i \mid 1 \leq i \leq 2^P\}$ and $\omega_i : \mathbb{R}^2 \to \mathbb{R}^2 : (x, y) \mapsto (\omega_i^x(x), \omega_i^y(y))$, where ω_i^x and ω_i^y are defined as follows:*

$$\omega_i^x(x) = \frac{1}{B^P}x + d_i^x \qquad\qquad \omega_i^y(y) = \frac{1}{B^P}y + f_\mathcal{P}(d_i^x) - \frac{f_\mathcal{P}(0)}{B^P}$$

Example 5.26 A logic program and two corresponding $IIFS_\mathcal{P}$s for $B = 4$:

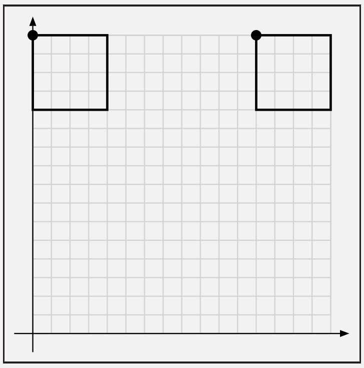
$\mathcal{P}$:

```
p(0).
p(s(X)) :- p(X).
p(X) :- not p(X).
```

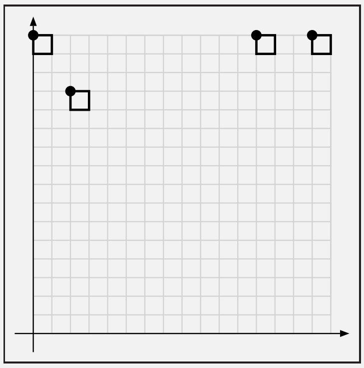
$|\cdot|$:

$|p(s^x(0))| = x + 1$

$IIFS_\mathcal{P}$ for $P = 1$



$IIFS_\mathcal{P}$ for $P = 2$



Each $IIFS_{\mathcal{P}}$ constructed for the program $\mathcal{P}$ and accuracy P corresponds to an $IFS^{l}_{\mathcal{P}}$ constructed for the periodicity P as in Algorithm 5.10. Therefore, we can use many results from Section 5.2.

Proposition 5.27 *Let $\{(\mathbb{R}^2, d_2), \Omega\}$ be an $IIFS_{\mathcal{P}}$ as constructed above. Then it is a hyperbolic iterated function system as introduced in Definition 3.31.*

Proof This proposition is an immediate consequence of Theorem 5.12. □

Proposition 5.28 *Let $\{(\mathbb{R}^2, d_2), \Omega\}$ be an $IIFS_{\mathcal{P}}$ with attractor A, as constructed above using the mapping R. And let $\pi_{\mathsf{x}}(\cdot)$ denote the projection onto the x-axis. Then $\pi_{\mathsf{x}}(A) = D_{\mathsf{f}}$, where D_{f} denotes the range of the mapping R.*

Proof This proposition is an immediate consequence of the Corollary 5.16. □

As this section will be fairly technical, a short outline is given now. First, a new complete metric space $(\mathcal{F}, d_{\mathsf{f}})$ is introduced, namely the space of continuous functions on D_{f}. For this space a contractive mapping T is given. As shown by Banach there is a uniquely determined fixed point for any such mapping. It is shown that the fixed point f is an interpolation function for the data constructed above. Moreover, it is shown that the sequence of functions $f_1, f_2, \ldots$ for increasing accuracies is a Cauchy sequence and hence converges. The final theorem of this section shows that this sequence of functions converges to the embedded $T_{\mathcal{P}}$-operator of the program $\mathcal{P}$.

Lemma 5.29 *Let $\mathcal{F}$ denote the set of continuous functions from D_{f} to $\mathbb{R}$ and let d_{f} be defined as $d_{\mathsf{f}}(f, g) = \max\{d_1(f(x), g(x)) \mid x \in D_{\mathsf{f}}\}$, where d_1 denotes the 1-dimensional Euclidean metric. Then $(\mathcal{F}, d_{\mathsf{f}})$ is a complete metric space.*

Proof This lemma follows immediately from Proposition 3.24, since D_{f} is a compact subset of $\mathbb{R}$. □

Lemma 5.30 *Let $\{(d_i^{\mathsf{x}}, d_i^{\mathsf{y}})\}$ be a set of interpolation data and $\{(\mathbb{R}^2, d_2), \Omega\}$ be an interpolating iterated function system with attractor A constructed from the program $\mathcal{P}$ using the accuracy P. Let $f_{\mathcal{P}}$ be the embedded $T_{\mathcal{P}}$-operator associated with the program $\mathcal{P}$ using the mapping R. Furthermore, let $(\mathcal{F}, d_f)$ be defined as in Lemma 5.29. Then $T_P : \mathcal{F} \to \mathcal{F}$ defined by $T_P(f)(x) = \omega_i^{\mathsf{y}} \circ f \circ \omega_i^{\mathsf{x}^{-1}}(x)$, where i is taken appropriately, is a contraction on $(\mathcal{F}, d_f)$ with contractivity factor $\frac{1}{B^P}$.*

Proof The proof is done in two steps. It is shown that T_P maps $\mathcal{F}$ into itself **(i)** and that it is a contraction **(ii)**. But first, we will have a closer look at $T_P f$.

$$T_P f(x) = T_{P,i} f(x) \qquad \text{for } x \in \left[d_i^{\mathsf{x}}, d_i^{\mathsf{x}} + \frac{R_{\mathsf{m}}}{B^P}\right] \cap D_{\mathsf{f}}$$

with each $T_{P,i} f$ defined as follows:

$$(T_{P,i} f)(x) = \omega_i^{\mathsf{y}}(f(\omega_i^{\mathsf{x}^{-1}}(x))) = \frac{1}{B^P} \cdot f\left((x - d_i^{\mathsf{x}}) \cdot B^P\right) + f_{\mathcal{P}}(d_i^{\mathsf{x}}) - \frac{f_{\mathcal{P}}(0)}{B^P}$$

For the sequel we will simply write $T_P f : D_{\mathsf{f}} \to \mathbb{R} : x \mapsto \omega_i^{\mathsf{y}}(f(\omega_i^{\mathsf{x}^{-1}}(x)))$ as Lemmas 5.14 – 5.17 allow to conclude that $T_P f$ is a well-defined function from D_{f} to $\mathbb{R}$.

(i) To show that T_P maps $\mathcal{F}$ into itself, we need to show that $T_P f : D_{\mathsf{f}} \to \mathbb{R}$ is a continuous function for all $f \in \mathcal{F}$. The continuity of $T_{P,i} f$ follows immediately from Proposition 3.19 as it is a composition of continuous functions. Furthermore, we need to show that for every sequence of points converging to a boundary point d_i^{x} and $d_i^{\mathsf{x}} + \frac{R_{\mathsf{m}}}{B^P}$, the sequence of images converges to the image of the boundary point. Notice that there is a gap between $\omega_i^{\mathsf{x}}(D_{\mathsf{f}})$ and $\omega_j^{\mathsf{x}}(D_{\mathsf{f}})$ for all $i \neq j$. This allows to assume that for every sequence $S_i = x_1, x_2, \ldots$ converging to d_i^{x} there is a $n \in \mathbb{N}$ such that $\mathsf{x}_j \in \{\omega_i^{\mathsf{x}}(x) \mid x \in D_{\mathsf{f}}\}$ for all $j > n$. Therefore, we can conclude that for $j > n$ we have $T_P f(x_j) = T_{P,i} f(x_j)$, which gives us that $d_1\left((T_{P,i} f)(x_j), (T_{P,i} f)(d_i^{\mathsf{x}})\right) = \left|f\left((x_j - d_i^{\mathsf{x}}) \cdot B^P\right) - f\left(0 \cdot B^P\right)\right|$. This is converging to 0 while x_j converges to d_i^{x}, since f is continuous.

(ii) The contractivity of T follows immediately from the definition

$$\begin{aligned}
d_{\mathsf{f}}(T_P f, T_P g) &= \max\{d_1(T_P f(x) - T_P g(x)) \mid x \in D_{\mathsf{f}}\} \\
&= \frac{1}{B^P} \cdot \max\left\{\left|f\left((x - d_i^{\mathsf{x}}) \cdot B^P\right) - g\left((x - d_i^{\mathsf{x}}) \cdot B^P\right)\right| \mid x \in D_{\mathsf{f}}\right\} \\
&\leq \frac{1}{B^P} \cdot d_{\mathsf{f}}(f, g)
\end{aligned}$$

Therefore, we can conclude that T_P is a contraction with contractivity factor $\frac{1}{B^P}$. $\square$

Theorem 5.31 *Let $D = \{(d_i^{\mathsf{x}}, d_i^{\mathsf{y}})\}$ be a sequence of interpolation data and $\{(\mathbb{R}^2, d_2), \Omega\}$ be an interpolating iterated function system with attractor A, as constructed in Definition 5.25 from the program $\mathcal{P}$ using accuracy P. Let $f_{\mathcal{P}}$ be the embedded $T_{\mathcal{P}}$-operator associated with the program $\mathcal{P}$ using the mapping R. Then there is a uniquely determined continuous function $f : D_{\mathsf{f}} \to \mathbb{R}$ with $T_P f = f$ which interpolates the data and for which the graph coincides with the attractor A.*

Proof Again the proof is done in two steps. First, we will show that the function f is uniquely determined and interpolates the data. Afterwards, we will show that the graph of this function coincides with the attractor.

(i) Lemma 5.30 tells us that the contractive mapping T_P maps $\mathcal{F}$ to itself. Using the contraction mapping principle as stated in Theorem 3.28 we can conclude that there is exactly one function f such that $T_P f = f$. This function is continuous since it is an element of $\mathcal{F}$. To show that f interpolates the data we need to show that $f(d_i^{\mathsf{x}}) = d_i^{\mathsf{y}} = f_{\mathcal{P}}(d_i^{\mathsf{x}})$ for all $(d_i^{\mathsf{x}}, d_i^{\mathsf{y}}) \in D$. Since we know that $T_P f = f$ we get

$$\begin{aligned} f(d_i^{\mathsf{x}}) &= \frac{1}{B^P} \cdot f\left((d_i^{\mathsf{x}} - d_i^{\mathsf{x}}) \cdot B^P\right) + f_{\mathcal{P}}(d_i^{\mathsf{x}}) - \frac{f_{\mathcal{P}}(0)}{B^P} \\ &= f_{\mathcal{P}}(d_i^{\mathsf{x}}) + \frac{f(0)}{B^P} - \frac{f_{\mathcal{P}}(0)}{B^P}. \end{aligned}$$

As there is a d_i^{x} which is equal to 0 we get $f(0) - \frac{f(0)}{B^P} = f_{\mathcal{P}}(0) - \frac{f_{\mathcal{P}}(0)}{B^P}$, which gives us the equality $f(0) = f_{\mathcal{P}}(0)$ and hence $f(d_i^{\mathsf{x}}) = f_{\mathcal{P}}(d_i^{\mathsf{x}})$ holds for all d_i^{x}.

(ii) To show that the graph $F = \{(x, f(x)) \mid x \in D_{\mathsf{f}}\}$ of the function f coincides with the attractor, it suffices to show that $F = \Omega(F)$. Since there is only one fixed point of Ω, it follows that $F = A$. Therefore, we will show **(ii.a)** $\Omega(F) \subseteq F$ and **(ii.b)** $F \subseteq \Omega(F)$. In order to prove **(ii.a)** we show that $\omega_i((x, f(x))) \in F$ for all $(x, f(x)) \in F$ and all $\omega_i \in \Omega$, i.e. $(\omega_i^{\mathsf{x}}(x), \omega_i^{\mathsf{y}}(f(x))) \in F$. This follows immediately from $f = T_P f = \omega_i^{\mathsf{y}} \circ f \circ \omega_i^{\mathsf{x}^{-1}}$, since we get that $f \circ \omega_i^{\mathsf{x}} = \omega_i^{\mathsf{y}} \circ f$ and hence $(\omega_i^{\mathsf{x}}(x), \omega_i^{\mathsf{y}}(f(x))) \in F$. Therefore, we can conclude that $\Omega(F) \subseteq F$. From Proposition 5.28 it follows that for all $x \in D_{\mathsf{f}}$ there is an $x' \in D_{\mathsf{f}}$ and an $\omega_i \in \Omega$ such that $x = \omega_i^{\mathsf{x}}(x')$. From $(x', f(x')) \in F$ and $f \circ \omega_i^{\mathsf{x}} = \omega_i^{\mathsf{y}} \circ f$ we can conclude that $f(x) = \omega_i^{\mathsf{y}}(f(x'))$and hence $\left(x, f(x)\right) = \left(\omega_i^{\mathsf{x}}(x'), \omega_i^{\mathsf{y}}(f(x'))\right)$. Therefore, **(ii.b)** holds which ends the proof. $\square$

Definition 5.32 (fractal interpolation function for $\mathcal{P}$) *The function f from Theorem 5.31 is called a* fractal interpolation function for $\mathcal{P}$ *with respect to P.*

Lemma 5.33 *Let f be a fractal interpolation function for a program $\mathcal{P}$ and some accuracy P. Then $f(x)$ is bounded by $f_{min} \leq f(x) = -\frac{R_m}{B^P-1}$ and $f_{max} = R_m + \frac{R_m}{B^P-1}$.*

Proof Using the definition of $\omega_i^{\mathsf{y}}(y) = \frac{y}{B^P} + f_{\mathcal{P}}(d_i^{\mathsf{x}}) - \frac{f_{\mathcal{P}}(0)}{B^P}$ a lower bound f_{min} for f can be constructed as follows:

$$
\begin{aligned}
f_{\mathsf{min}_1} &= \frac{0}{B^P} - \frac{R_{\mathsf{m}}}{B^P} \\
f_{\mathsf{min}_2} &= \frac{f_{\mathsf{min}_1}}{B^P} - \frac{R_{\mathsf{m}}}{B^P} = \frac{R_{\mathsf{m}}}{(B^P)^2} - \frac{R_{\mathsf{m}}}{B^P} \\
f_{\mathsf{min}_n} &= -\sum_{i=1}^{n} \frac{R_{\mathsf{m}}}{(B^P)^i} \\
f_{\mathsf{min}} &= \lim_{n\to\infty} f_{\mathsf{min}_n} = -\frac{R_{\mathsf{m}}}{B^P - 1}
\end{aligned}
$$

Analogously f_{max} is constructed with $f_{\mathsf{max}} = R_{\mathsf{m}} + \frac{R_{\mathsf{m}}}{B^P-1}$. □

Lemma 5.34 *Let $\{\mathbb{R}^2, \Omega\}$ be an IIFS$_{\mathcal{P}}$ constructed for an acyclic program $\mathcal{P}$ and some accuracy i. Let f_i denote the corresponding fractal interpolation function. Then the sequence $f_1, f_2, \ldots$ is a Cauchy sequence on $(\mathcal{F}, d_{\mathsf{f}})$ as introduced in Lemma 5.29.*

Proof The proof is done in two steps. First, it is shown that the distance between f_i and f_{i+1} is bounded and converging to 0. Afterwards, it is shown that the sequence is Cauchy.

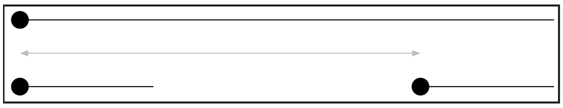
(i) This part of the proof is concerned with the distance between the two fractal interpolation functions f_i and f_{i+1}. For readability reasons f is used for f_i and $\hat{f}$ is used for f_{i+1}. For both functions $Tf = f$ and $\hat{T}\hat{f} = \hat{f}$ hold, where T and $\hat{T}$ denote the operators as introduced in Lemma 5.30, constructed for the accuracies $P = i$ and $\hat{P} = i + 1$ respectively. Furthermore, d_j^{x} denotes the offset of the j-th mapping used to construct T and $\hat{d}_j^{\mathsf{x}}$ respectively for $\hat{T}$. From $Tf = f$ and $\hat{T}\hat{f} = \hat{f}$ we can conclude that

$$
\begin{aligned}
f(x) &= \frac{f\left((x - d_j^{\mathsf{x}}) \cdot B^i\right)}{B^i} + f_{\mathcal{P}}(d_j^{\mathsf{x}}) - \frac{f_{\mathcal{P}}(0)}{B^i} && \text{for } x \in \left[d_i^{\mathsf{x}}, d_i^{\mathsf{x}} + \frac{R_{\mathsf{m}}}{B^i}\right] \cap D_{\mathsf{f}} \\
\hat{f}(x) &= \frac{\hat{f}\left((x - \hat{d}_j^{\mathsf{x}}) \cdot B^{i+1}\right)}{B^{i+1}} + f_{\mathcal{P}}(\hat{d}_j^{\mathsf{x}}) - \frac{f_{\mathcal{P}}(0)}{B^{i+1}} && \text{for } x \in \left[\hat{d}_i^{\mathsf{x}}, \hat{d}_i^{\mathsf{x}} + \frac{R_{\mathsf{m}}}{B^{i+1}}\right] \cap D_{\mathsf{f}}
\end{aligned}
$$

Therefore, we get for the distance $d_{\mathsf{f}}(f,\hat{f})$:

$$
\begin{aligned}
d_{\mathsf{f}}(f,\hat{f}) &= \max_x \Bigg\{ \Bigg| \left(\frac{f\big((x-d^{\mathsf{x}}_j)\cdot B^i\big)}{B^i} + f_{\mathcal{P}}(d^{\mathsf{x}}_j) - \frac{f_{\mathcal{P}}(0)}{B^i} \right) - \\
&\qquad\qquad \left(\frac{\hat{f}\big((x-\hat{d}^{\mathsf{x}}_k)\cdot B^{i+1}\big)}{B^{i+1}} + f_{\mathcal{P}}(\hat{d}^{\mathsf{x}}_k) - \frac{f_{\mathcal{P}}(0)}{B^{i+1}} \right) \Bigg| \Bigg\} \\
&\le \max_x \Bigg\{ \left| \frac{B\cdot f\big((x-d^{\mathsf{x}}_j)\cdot B^i\big) - \hat{f}\big((x-\hat{d}^{\mathsf{x}}_k)\cdot B^{i+1}\big)}{B^{i+1}} \right| + \\
&\qquad\qquad \left| f_{\mathcal{P}}(d^{\mathsf{x}}_j) - f_{\mathcal{P}}(\hat{d}^{\mathsf{x}}_k) \right| + \left| \frac{-B\cdot f_{\mathcal{P}}(0) + f_{\mathcal{P}}(0)}{B^{i+1}} \right| \Bigg\} \\
&\le \max_x \left\{ \frac{B\cdot (R_{\mathsf{m}} + \frac{R_{\mathsf{m}}}{B^i-1}) + \frac{R_{\mathsf{m}}}{B^{i+1}-1}}{B^{i+1}} + \frac{1}{B-2}\cdot \left| d^{\mathsf{x}}_j - \hat{d}^{\mathsf{x}}_k \right| + \frac{(B-1)\cdot R_{\mathsf{m}}}{B^{i+1}} \right\}
\end{aligned}
$$

The last step uses the fact that $f_{\mathcal{P}}$ is a contraction on $(\mathbb{R}, d_1)$ with contractivity factor $\frac{1}{B-2}$ and the results of Lemma 5.33 concerning minima and maxima of f_i.
Since d^{x}_j and $\hat{d}^{\mathsf{x}}_k$ are chosen with respect to the same x we can conclude that $\hat{d}^{\mathsf{x}}_k = d^{\mathsf{x}}_j$ or $\hat{d}^{\mathsf{x}}_k = d^{\mathsf{x}}_j + \frac{R_{\mathsf{m}}}{B^i}\cdot\frac{B-1}{B}$, as depicted in the following picture:



The upper line depicts the "image" of D_{f} with respect to ω^{x}_j, hence has a length of $\frac{R_{\mathsf{m}}}{B^i}$, whereas the lower lines have a length of $\frac{R_{\mathsf{m}}}{B^{i+1}}$.
Therefore, the maximal distance between d^{x}_j and $\hat{d}^{\mathsf{x}}_k$ is bounded by $\frac{R_{\mathsf{m}}}{B^i}\cdot\frac{B-1}{B}$ and we can conclude

$$
\begin{aligned}
d_{\mathsf{f}}(f,\hat{f}) &\le \frac{B\cdot (R_{\mathsf{m}} + \frac{R_{\mathsf{m}}}{B^i-1}) + \frac{R_{\mathsf{m}}}{B^{i+1}-1}}{B^{i+1}} + \frac{1}{B-2}\cdot\frac{R_{\mathsf{m}}}{B^i}\cdot\frac{B-1}{B} + \frac{(B-1)\cdot R_{\mathsf{m}}}{B^{i+1}} \\
&\le \frac{B\cdot (R_{\mathsf{m}} + \frac{R_{\mathsf{m}}}{B^i-1}) + \frac{R_{\mathsf{m}}}{B^{i+1}-1} + \frac{1}{B-2}\cdot R_{\mathsf{m}}\cdot(B-1) + (B-1)\cdot R_{\mathsf{m}}}{B^{i+1}} \\
&\le R_{\mathsf{m}}\cdot \frac{B\cdot (1 + \frac{1}{B^i-1}) + \frac{1}{B^{i+1}-1} + \frac{1}{B-2}\cdot(B-1) + (B-1)}{B^{i+1}} \\
&\le \frac{R_{\mathsf{m}}}{B^{i+1}}\cdot\left(2B + \frac{B}{B^i-1} + \frac{1}{B^{i+1}-1} + \frac{B-1}{B-2} - 1\right) < \frac{R_{\mathsf{m}}}{B^{i+1}}\cdot 5B = \frac{5R_{\mathsf{m}}}{B^i}
\end{aligned}
$$

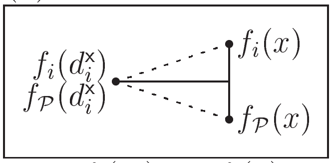
(ii) Knowing that $d_{\mathsf{f}}(f_i, f_{i+1}) \le \frac{5R_{\mathsf{m}}}{B^i}$ we can conclude (assuming $j \le k$) that

$$
d_{\mathsf{f}}(f_j, f_k) = \sum_{i=j}^{k} \frac{5R_{\mathsf{m}}}{B^i} = \frac{5R_{\mathsf{m}}}{B-1}\left(\frac{1}{B^j} - \frac{1}{B^k}\right)
$$

For a fixed j we get that the value of $d_{\mathsf{f}}(f_j, f_k)$ is bounded with $d_{\mathsf{f}}(f_j, f_k) \le \frac{5R_{\mathsf{m}}}{B-1}\cdot\frac{1}{B^j}$. Therefore, we can conclude that $f_1, f_2, \ldots$ is a Cauchy sequence, since for any $\epsilon > 0$ there is a n such that for all $j, k > n$ we have $d_{\mathsf{f}}(f_j, f_k) < \epsilon$. $\square$

Theorem 5.35 *Let* $\{(\mathbb{R}^2, d_2), \Omega\}$ *be an* $IIFS_{\mathcal{P}}$ *constructed for an acyclic program* $\mathcal{P}$ *and some accuracy* P *as introduced in Definition 5.25. Let* $D = \{(d_i^{\mathsf{x}}, d_i^{\mathsf{y}})\}$ *denote the corresponding set of interpolation data. Let* $f_{\mathcal{P}}$ *be the embedded* $T_{\mathcal{P}}$*-operator associated with* $\mathcal{P}$ *using the mapping* R*. Then the sequence of fractal interpolation functions* f_i *for the accuracies* $0 < i$ *converges uniformly to* $f_{\mathcal{P}}$ *in the complete metric space* $(\mathcal{F}, d_{\mathsf{f}})$*.*

Proof For this proof we need to notice that for any $\epsilon > 0$ there is an accuracy P such that for all $x \in D_{\mathsf{f}}$ we have $d_1(x, d_i^{\mathsf{x}}) < \epsilon$, where d_i^{x} is the offset of the appropriate chosen mapping ω_i^{x} of the $IIFS_{\mathcal{P}}$ constructed for accuracy P. Therefore, we can construct a sequence $d_1^{\mathsf{x}}, d_2^{\mathsf{x}}, \ldots \to x$ for all $x \in D_{\mathsf{f}}$, with d_i^{x} now denoting the data point of the i-th interpolation function which is closest to x.
From the continuity of f_i and the uniform convergence of f_i to f we can conclude that the sequence $f_1(d_1^{\mathsf{x}}), f_2(d_2^{\mathsf{x}}), \ldots$ converges to $f(x)$. Knowing that the f_i are interpolation functions, hence $f_i(d_i^{\mathsf{x}}) = f_{\mathcal{P}}(d_i^{\mathsf{x}})$ we can conclude that $f_{\mathcal{P}}(d_1^{\mathsf{x}}), f_{\mathcal{P}}(d_2^{\mathsf{x}}), \ldots \to f(x)$. Finally taking into account that $f_{\mathcal{P}}$ is continuous for acyclic programs, we can conclude that $f_{\mathcal{P}}(d_1^{\mathsf{x}}), f_{\mathcal{P}}(d_2^{\mathsf{x}}), \ldots \to f_{\mathcal{P}}(x)$ and hence $f(x) = f_{\mathcal{P}}(x)$ for all $x \in D_{\mathsf{f}}$.
The figure depicts the points involved in the proof. Since we want to estimate the distance between $f_i(x)$ and $f_{\mathcal{P}}(x)$ we apply the triangular inequality and get $d_1(f_i(x), f_{\mathcal{P}}(x)) \leq d_1(f_i(x), f_i(d_i^{\mathsf{x}})) + d_1(f_{\mathcal{P}}(x), f_{\mathcal{P}}(d_i^{\mathsf{x}}))$.



Knowing that f_i and $f_{\mathcal{P}}$ are continuous, we can conclude that $f_i(d_i^{\mathsf{x}}) \to f_i(x)$ and $f_{\mathcal{P}}(d_i^{\mathsf{x}}) \to f_{\mathcal{P}}(x)$ while $d_i^{\mathsf{x}} \to x$. Furthermore, $f(x) = f_{\mathcal{P}}(x)$ for all $x \in D_{\mathsf{f}}$, since $f_i \to f$ and for all d_i^{x} we have $f_{\mathcal{P}}(d_i^{\mathsf{x}}) = f_i(d_i^{\mathsf{x}})$. □

This theorem shows that we can approximate the graph of an acyclic logic program arbitrarily well. Unfortunately, the necessary number of mappings grows exponentially with the accuracy.

6. Applications, Results & Further Work

This section shows possible applications of the results achieved so far. At the end, the main results are repeated and some ideas for further work are presented.

6.1. Applications

Three different ideas are presented to use the theoretic results of this thesis. First, some areas of the plot are related to different types of models, which were introduced in Section 2.2.3. Afterwards, two possible embeddings into (recurrent) neural networks are presented. Everything discussed here is only briefly described and only rough ideas are given. The technical details remain to be worked out in the future.

6.1.1. Where are the Models in the Graph

From the way R is defined we can draw some conclusions concerning the relation of interpretations and their associated real values. Some of them are shown in Figure 12.

$$
\begin{array}{ccc}
T_{\mathcal{P}}(I) \subseteq I & \Rightarrow & R(T_{\mathcal{P}}(I)) \leq R(I) \\
T_{\mathcal{P}}(I) \supseteq I & \Rightarrow & R(T_{\mathcal{P}}(I)) \geq R(I) \\
T_{\mathcal{P}}(I) = I & \Rightarrow & R(T_{\mathcal{P}}(I)) = R(I)
\end{array}
$$

Figure 12: Relations between interpretations, their consequences and their real values

Furthermore, we can draw conclusions concerning the (pre-)fixed points of the $T_{\mathcal{P}}$-operator. As discussed in Section 2.2.3, models are pre-fixed points of the $T_{\mathcal{P}}$-operator. Knowing that prefixed points fulfil $T_{\mathcal{P}}(I) \subseteq I$ we can conclude that for all models I we have $R(T_{\mathcal{P}}(I)) \leq R(I)$. This corresponds to the triangular area shown in Figure 13, i.e. all models "live" in the area below the diagonal.

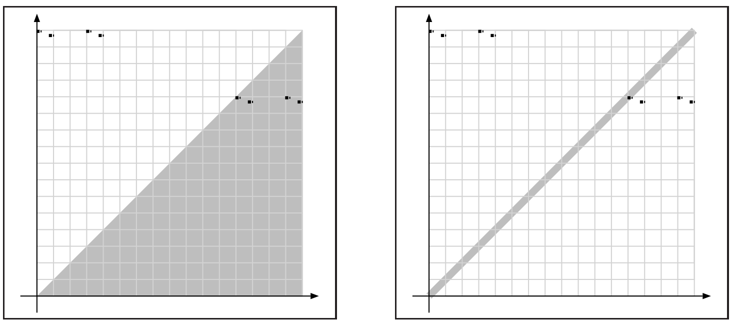


Figure 13: Interesting regions in the graphs of logic programs

Stable models are defined as fixed points of the $T_{\mathcal{P}}$-operator, i.e. $T_{\mathcal{P}}(I) = I$. Therefore, $R(T_{\mathcal{P}}(I)) = R(I)$ holds for all models I, which coincides with the line from $(0, 0)$ to $(R_{\mathsf{m}}, R_{\mathsf{m}})$ as shown in the Figure.

Unfortunately, there is no way known to compute the intersection of an attractor and the diagonal or the triangle below it, but maybe there are fast "graphical" algorithms which can approximate those intersections. In [Dub95] the converse was shown, namely that for general IFSs this intersection cannot be computed. Taking into account that the IFSs constructed here are all of a certain type this might be possible and remains to be worked out in the future.

6.1.2. Approximation of $f_{\mathcal{P}}$ with Radial Basis Function Networks

Radial Basis Function (RBF) are functions for which the output is symmetric around a centre, as shown in Figure 14. These functions can usually be described using their centre, width and height as depicted and are denoted ϕ in the sequel.

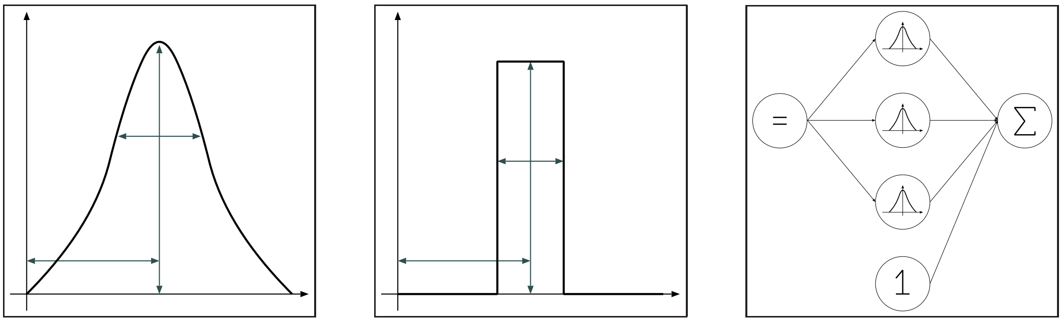


Figure 14: Two examples of radial basis functions and a RBF-network

RBFs are known to be universal approximators, i.e. linear combinations of the form $y(\mathbf{x}) = \sum_{i=1}^{n} w_i \cdot \phi_i(\mathbf{x})$ can be used to approximate (almost) any given function [DM90, JS93]. Radial Basis Function Networks (RBFNs) are connectionistic implementations of these linear combinations, as shown in the figure above, consisting of three layers [SB90, PG90]. Units from the input layer simply "forward" their input, i.e. $f_{\mathsf{in}}(x) = x$. The hidden layer-units compute radial basis functions, i.e. $f_{\mathsf{hid}_i}(x) = \phi_i(x)$. Furthermore, a so-called bias-unit with $f_{\mathsf{b}} = 1$ is added to the hidden layer to compensate some global offset. Units from the output layer sum up all their inputs and return the weighted sum $f_{\mathsf{out}} = \sum_{i=0}^{n} w_i \cdot x_i$, where the 0-th weight is the bias, since $x_0 = f_b = 1$. The connections between input and hidden layer have all weight 1 and the connections from the hidden to the output layer contain the weights w_i.

As mentioned above, such linear combinations can approximate (almost) any given function. Without going into the mathematical details this includes all continuous functions on compact subsets of $\mathbb{R}$. Here we will use RBFNs to approximate $f_{\mathcal{P}}$, which is a continuous function on a compact set as shown in Section 4.3.

The units used for this approach compute piecewise linear functions as defined in Figure 16. Using these RBF_x^y-units we can construct a network which computes an approximation of $f_{\mathcal{P}}$ for a given program $\mathcal{P}$.

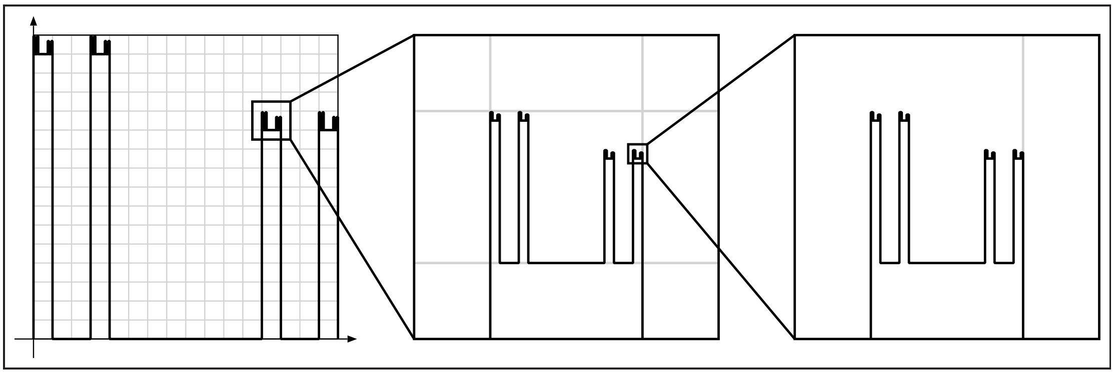


Figure 15: The embedded $T_{\mathcal{P}}$-operator from Example 5.23

$$f_{\mu,\sigma,h}(x) = \begin{cases} h & \text{if } |x-\mu| \leq \sigma \\ 0 & \text{otherwise} \end{cases}$$

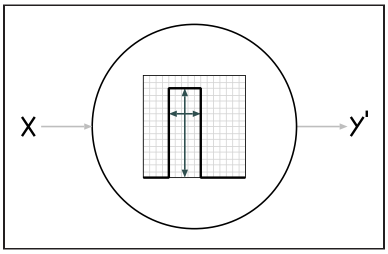


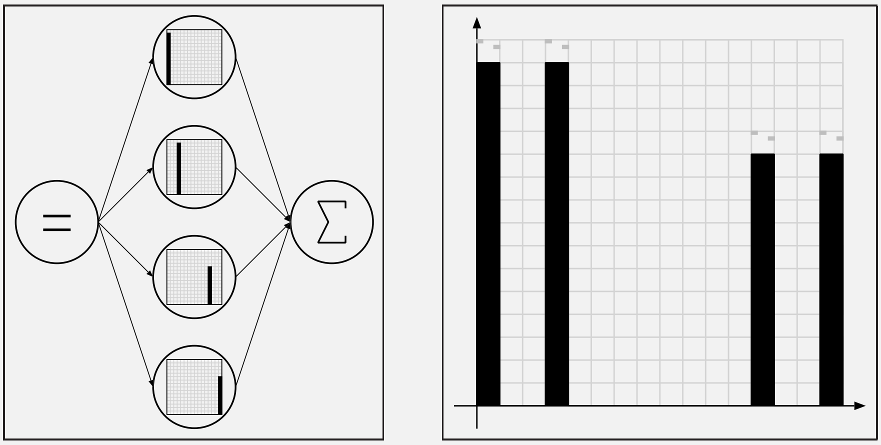
Figure 16: Dynamics and scheme of an RBF^y_x-unit

Algorithm 6.1 (Construction of $RBFN_{\mathcal{P}}$)
Let $\mathcal{P}$ be a logic program and B the base of the embedding R.

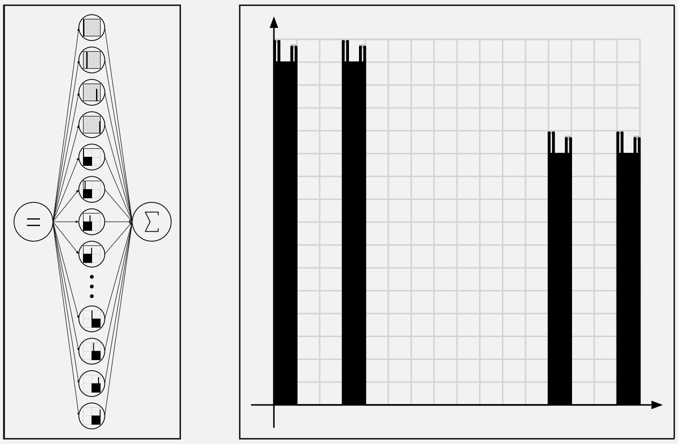
1. Choose a periodicity $P \geq 1$ and a level $L \geq 1$.
2. Create the set $N_1 = \{RBF^y_{x,i} \mid 1 \leq i \leq P\}$ as follows:
   a) Compute the $IFS^l_{\mathcal{P}} = \{(\mathbb{R}^2, d_2), \Omega\}$ for $\mathcal{P}$ using the periodicity P, with $\Omega = \{(\omega^{\mathsf{x}}_i, \omega^{\mathsf{y}}_i) \mid 1 \leq i \leq P\}$, $\omega^{\mathsf{x}}_i(x) = \frac{1}{B^P} \cdot x + d^{\mathsf{x}}_i$ and $\omega^{\mathsf{y}}_i(y) = \frac{1}{B^P} \cdot y + d^{\mathsf{y}}_i$ as described in Algorithm 5.10.
   b) Add for all i the $RBF^y_{x,i}$-unit with $\sigma_i = \frac{1}{2 \cdot B^P}$, $\mu_i = d^{\mathsf{x}}_i + \sigma$ and $h_i = d^{\mathsf{y}}_i$ to N_1.
3. Create an empty 3-layered RBF-network. Add one input unit to the input layer and one output unit to the output layer.
4. For each level $1 \leq l \leq L$ do ...
   a) Replace the current hidden layer with P copies of itself.
   b) Multiply μ, σ and h of all copied units with $\frac{1}{P^{l-1}}$.
   c) Add to each μ of the i-th copy d^{x}_i from Step 2.a.
   d) Add N_1 to the hidden layer.

Unfortunately, the number of units grows exponentially with the level L, since the hidden layer contains $\sum_{i=1}^{L-1} P^i$ units. The following examples show RBF networks for the program from Example 5.23. Using the periodicity $P = 2$ they are constructed for level $L = 1$ and $L = 2$ as described in Algorithm 6.1.

Example 6.2 (An $RBFN_{\mathcal{P}}$ for $P = 2$ and $L = 1$)



Example 6.3 (An $RBFN_{\mathcal{P}}$ for $P = 2$ and $L = 2$)



6.1.3. Approximation of $f_{\mathcal{P}}$ using Recurrent Radial Basis Function Networks

As mentioned in the previous section, the number of units necessary grows exponentially with the accuracy (level) of the network. Therefore, another approach is presented here which converts the RBFN into a recurrent one with a very limited number of units.

To simplify the network a new type of unit is introduced here, denoted $RBF_{s,x}^{x,y}$-unit. It computes two distinct output-values x' and y'. Furthermore, it computes a parametrised radial basis function, where an additional scaling s is applied to y'. These units can be understood as abbreviations, since they can be converted into a network consisting of simple units. Figure 17 shows the dynamics and a schematic plot of an $RBF_{s,x}^{x,y}$-unit.

$$x' = \begin{cases} (x - \mu + \sigma) \cdot B^P & \text{if } |\mu - x| \leq \sigma \\ 0 & \text{otherwise} \end{cases}$$

$$y' = \begin{cases} s \cdot h & \text{if } |\mu - x| \leq \sigma \\ 0 & \text{otherwise} \end{cases}$$

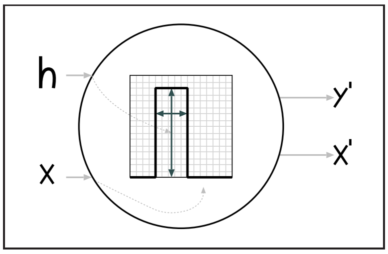


Figure 17: Dynamics and scheme of an $RBF_{s,x}^{x,y}$-unit

Using $RBF_{s,x}^{x,y}$-units it is possible to construct a network as shown in Figure 18 using the following algorithm.

Algorithm 6.4 (Construction of recurrent $RBFN_{\mathcal{P}}$)
Let $\mathcal{P}$ be a logic program and B the base of the embedding R.

1. Choose a periodicity $P \geq 1$.
2. Create an empty 3-layered RBF-network. Add three input units (s, x, y) to the first layer and three output units (s', x', y') to the third. As mentioned in the previous section, the input units compute the identity function and the output units return a weighted sum of their inputs.
3. The hidden layer consists of P $RBF_{s,x}^{x,y}$-units initialised as follows:
   a) Compute the $IFS_{\mathcal{P}}^{t} = \{(\mathbb{R}^2, d_2), \Omega\}$ for $\mathcal{P}$ using the periodicity P, with $\Omega = \{(\omega_i^{\mathsf{x}}, \omega_i^{\mathsf{y}}) \mid 1 \leq i \leq P\}$, $\omega_i^{\mathsf{x}}(x) = \frac{1}{B^P} \cdot x + d_i^{\mathsf{x}}$ and $\omega_i^{\mathsf{y}}(y) = \frac{1}{B^P} \cdot y + d_i^{\mathsf{y}}$, as described in Algorithm 5.10.
   b) For all i the $RBF_{s,x,i}^{x,y}$-unit is initialised with $\sigma_i = \frac{1}{2 \cdot B^P}$, $\mu_i = d_i^{\mathsf{x}} + \sigma$ and $h_i = d_i^{\mathsf{y}}$.
4. Connect the units as shown in Figure 18, where all weights are set to 1, but the connection from s to s' is set to $1/B^P$.

The network computes an approximation of $f_{\mathcal{P}}$ for a given input $R(I)$. The three inputs s_0, y_0 and x_0 need to be initialised with $s_0 = 1$, $y_0 = 0$ and $x_0 = R(I)$. Each iteration through the network performs the following computations:

- The scaling factor s is multiplied with $\frac{1}{B^P}$.
- Each $RBF^{x,y}_{s,x}$-unit computes the corresponding outputs x' and y', where for exactly one unit $x', y' \neq 0$. Since x_0 was initialised with $R(I)$, the output of the "active" $RBF^{x,y}_{s,x}$-units after the first iteration is $x' = (x - \mu + \sigma) \cdot B^P$ and, therefore, we have $x' = (x - d^{\mathsf{x}}_i) \cdot B^P = R\big(R^{-1}(x) \setminus R^{-1}(d^{\mathsf{x}}_i)\big) \cdot B^P$. This coincides with a "zooming into the interpretation", i.e. x' is a left-shifted version of x.
- The current y'-output of the "active" unit is added to the previous y.

Taking a closer look at the construction we should find the output y' converging to $f_{\mathcal{P}}(x_0)$, but the details remain to be worked out in the future.

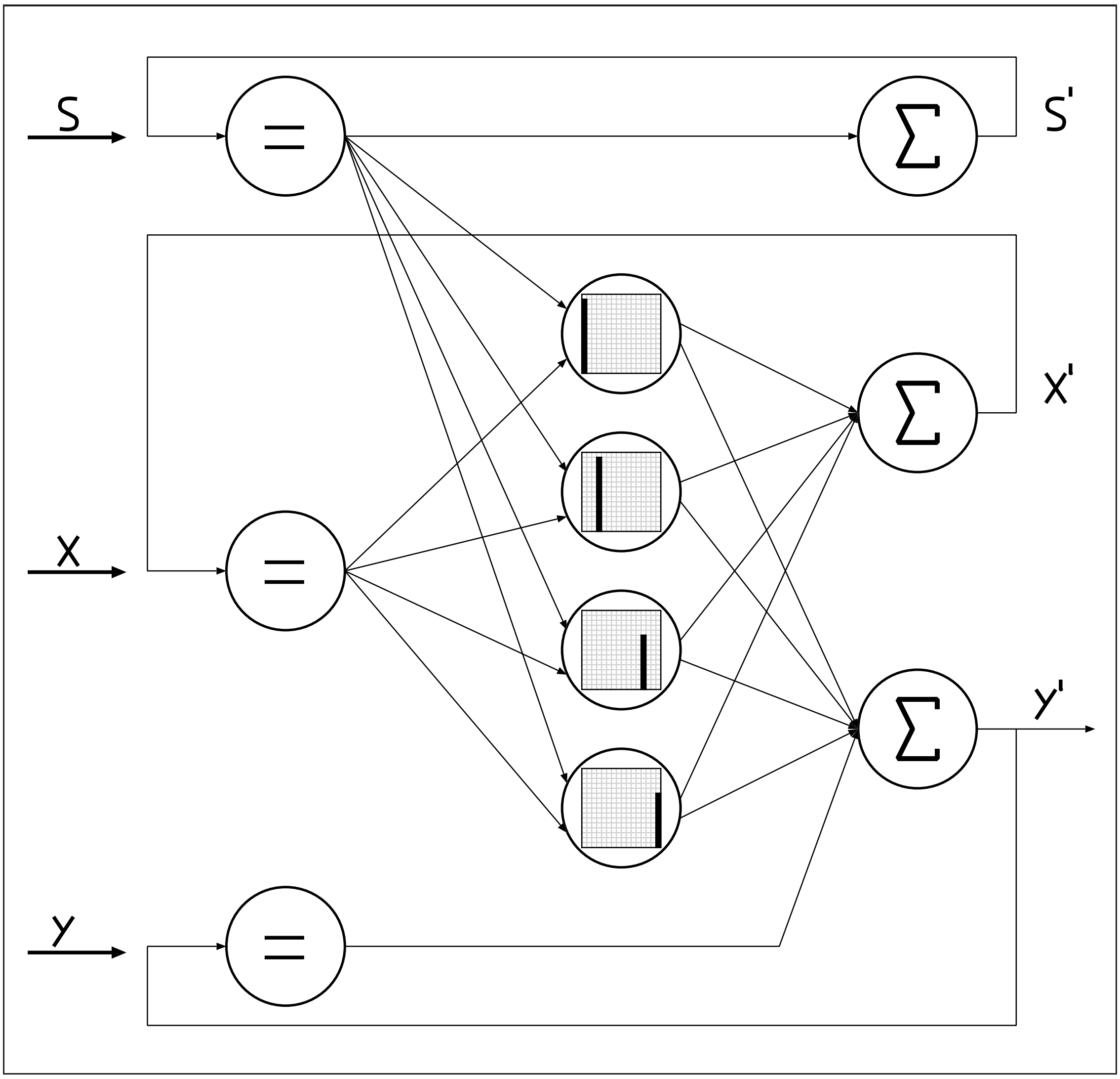


Figure 18: Recurrent $RBF^{x,y}_{s,x}$-network

6.2. Results

This section briefly repeats the main results of this thesis.

Properties of R Within Section 4.1 the mapping proposed by Hölldobler, Kalinke and Störr in [HKS99] was discussed, generalised to arbitrary bases > 2 and important properties shown. Furthermore, the right-shift operator $\dot{\rightarrow}$ was defined, which helped to prove later parts of the thesis.

A generalised version of the Fitting metric ... was defined in Section 4.3. It was shown that the proposed generalised function is an ultra metric on the space of interpretations and that the results achieved by Fitting concerning the contractivity of certain $T_{\mathcal{P}}$-operators, generalise to this version. Furthermore, it was shown that the metric is equivalent to the natural metric on $\mathbb{R}$ and hence the contractivity of $T_{\mathcal{P}}$ can be carried over to the generalised $f_{\mathcal{P}}$ (for bases > 3).

Fractal nature of logic programs This section established the promised connection between logic programs and iterated function systems. The Equivalence Theorem gives necessary and sufficient conditions under which the graph of the program and the attractor of an IFS coincide. Using the theorem it was shown that the graph of a program is a fractal, if the associated $T_{\mathcal{P}}$-operator is continuous. Since it is continuous for all programs with finite $ground(\mathcal{P})$, this shows that the self-similarity found in the pictures is not an artefact, but rather due to the deep underlying connection. Furthermore, a method was proposed to construct a linear IFS for a given logic program and it was shown that this construction is valid for a certain subclass of programs. Finally, the idea of fractal interpolation was used to generalise the results in term of arbitrary well approximations.

Possible applications In Section 6.1 possible applications of the theoretic results were presented. This includes an embedding of the constructed iterated function systems into recurrent neural networks and hence closing the circle to the original motivation.

6.3. Further Work

Other embeddings Throughout this thesis one particular embedding was used, namely the mapping R introduced in [HKS99]. But other mappings might also exist and could lead to new results.

Other consequence operators Within this thesis only the $T_{\mathcal{P}}$-operator associated with a logic program was taken into account. But there are other operators like e.g. $\Phi_{\mathcal{P}}$ for which similar results might hold.

Extension to other classes of logic programs The last example presented in appendix B shows that all restrictions made concerning the classes of programs are too strong. It shows a non-acyclic logic program which is neither linear nor separable. Still the plot of the graph shows some pattern which could indicate a self-similarity. Therefore, the quest is still open and it remains to be worked out in the future for which classes of logic programs the graphs of the $T_{\mathcal{P}}$-operator can be approximated using iterated function systems.

Generalisation Within this thesis a link between logic programs and iterated function systems was established. But logic programs are only one instance of more general theories like, for example, the recursion theory. Therefore, the study of those more general theories with respect to their link to iterated function systems could be fruitful.

Dynamical systems and iterations of $T_{\mathcal{P}}$ To any totally disconnected iterated function system a dynamical system can be assigned, which is called the *shift dynamical system*. Using the coincidence of the graph of the program and the attractor of the IFS one can relate this dynamical system to iterations of the $T_{\mathcal{P}}$-operator. Therefore, results known from one theory might carry over to the other.

Reversion of the construction and inductive logic programming This thesis is concerned with the construction of iterated function systems for a given program $\mathcal{P}$. The study of a reverse procedure might strengthen the link and provide new insights into the underlying relation. Unfortunately, it is still not clear whether such reverse construction exists. Inductive logic programming is concerned with the automatic construction of logic programs for a given set of examples such that the program explains the data. Using the embedding R this data can be plotted as a graph in $\mathbb{R}^2$. Techniques known from fractal image compression allow the construction of IFSs such that the attractors are similar to the plot. Applying a reverse construction we get a logic program with a graph coinciding with the attractor. Therefore, the constructed logic program is an explanation for the data.

A. Attractors of Famous Iterated Function Systems

The following plots were created using Glito, a free IFS creator, which can be found at `http://emmanuel.debanne.free.fr/glito/#english`. The examples below contain the mappings of the IFS, the complete attractor and a "zooming"-sequence showing their self-similar structure.

Cantor set $\mathcal{C}$

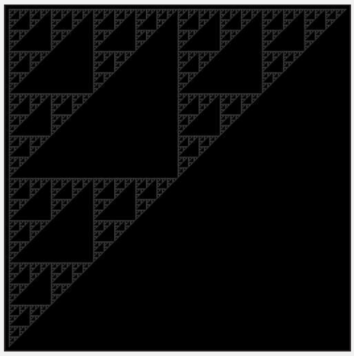
$$\omega_1(x) = \frac{1}{3}x \qquad \omega_2(x) = \frac{1}{3}x + \frac{1}{3}$$

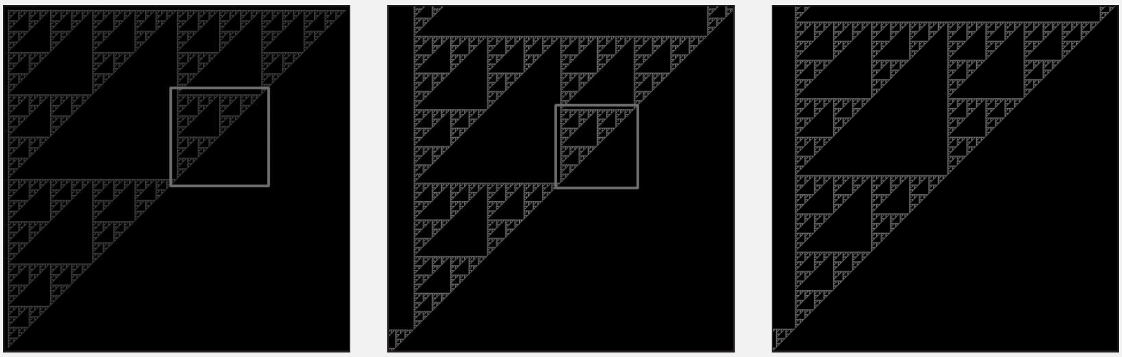
Sierpinski triangle

$$\omega_1(x,y) = \begin{pmatrix} .5 & 0 \\ 0 & .5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -0.25 \\ -0.25 \end{pmatrix}$$

$$\omega_2(x,y) = \begin{pmatrix} .5 & 0 \\ 0 & .5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -0.25 \\ 0.25 \end{pmatrix}$$

$$\omega_2(x,y) = \begin{pmatrix} .5 & 0 \\ 0 & .5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0.25 \\ 0.25 \end{pmatrix}$$

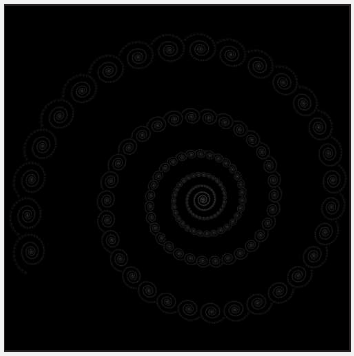


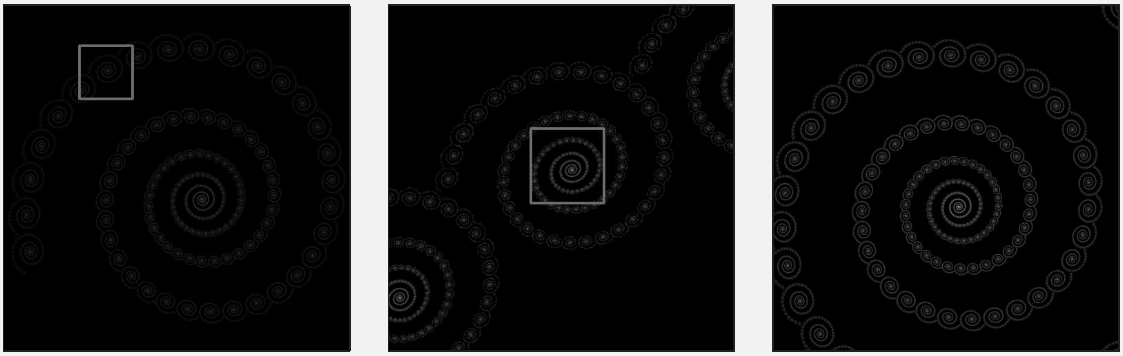


A fractal spiral

$$\omega_1(x,y) = \begin{pmatrix} 0.06 & -0.09 \\ 0.1 & 0.07 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -0.43 \\ -0.13 \end{pmatrix}$$

$$\omega_2(x,y) = \begin{pmatrix} 0.96 & 0.2 \\ -0.2 & 0.96 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$





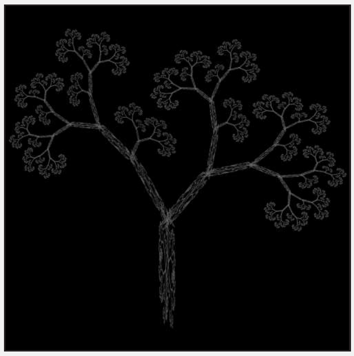
A fractal tree

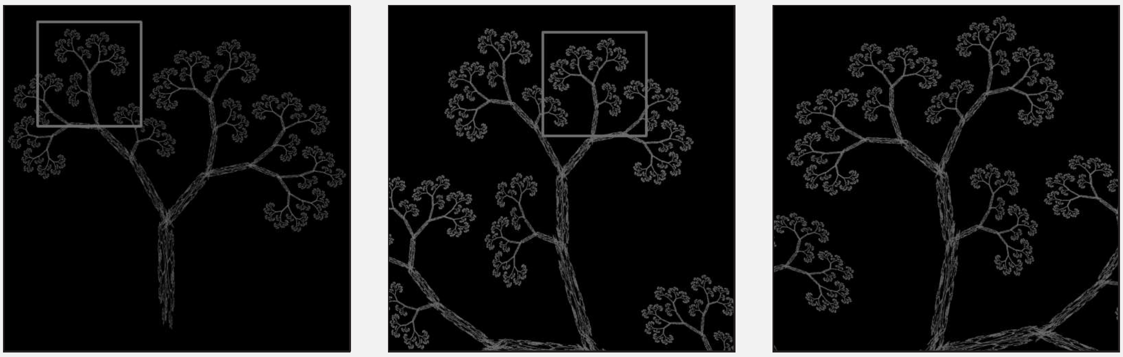
$$\omega_1(x,y) = \begin{pmatrix} 0.05 & 0 \\ 0 & 0.4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -0.06 \\ -0.47 \end{pmatrix}$$

$$\omega_2(x,y) = \begin{pmatrix} 0.03 & -0.14 \\ 0 & 0.26 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -0.16 \\ -0.01 \end{pmatrix}$$

$$\omega_3(x,y) = \begin{pmatrix} 0.56 & 0.44 \\ -0.37 & 0.51 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0.3 \\ 0.15 \end{pmatrix}$$

$$\omega_4(x,y) = \begin{pmatrix} 0.19 & 0.07 \\ -0.1 & 0.15 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -0.2 \\ 0.28 \end{pmatrix} \qquad \ldots$$





A fractal fern

$$\omega_1(x,y) = \begin{pmatrix} 0.01 & 0.002 \\ 0 & 0.2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\omega_2(x,y) = \begin{pmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0.16 \end{pmatrix}$$

$$\omega_3(x,y) = \begin{pmatrix} 0.087 & -0.26 \\ 0.25 & 0.22 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0.004 \\ 0.09 \end{pmatrix}$$

$$\omega_4(x,y) = \begin{pmatrix} -0.14 & 0.28 \\ 0.14 & 0.24 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0.125 \end{pmatrix}$$

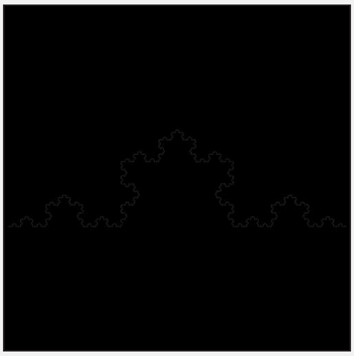
The Koch curve (snowflake)

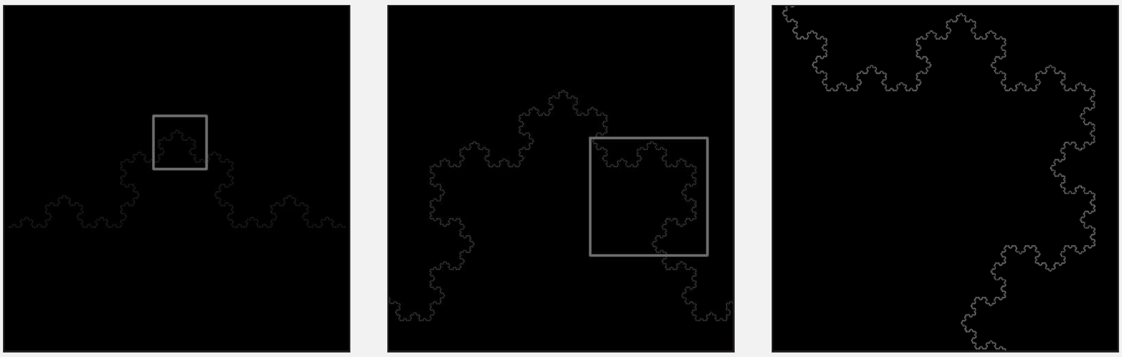
$$\omega_1(x,y) = \begin{pmatrix} 0.33 & 0 \\ 0 & 0.33 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -0.33 \\ 0 \end{pmatrix}$$

$$\omega_2(x,y) = \begin{pmatrix} 0.33 & 0 \\ 0 & 0.33 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0.33 \\ 0 \end{pmatrix}$$

$$\omega_3(x,y) = \begin{pmatrix} 0.17 & -0.3 \\ 0.29 & 0.17 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -0.08 \\ 0.14 \end{pmatrix}$$

$$\omega_4(x,y) = \begin{pmatrix} 0.17 & 0.3 \\ -0.3 & 0.17 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0.08 \\ 0.14 \end{pmatrix}$$





B. Examples of Transformed Logic Programs

This section presents some logic programs, their graphs and corresponding iterated function systems. More examples can be found at `http://www.louise15.de/LP2IFS`, where an applet is made available to create own plots.

Each example shows the program, the level mappings and the $IFS_{\mathcal{P}}$ constructed using Algorithm 5.10. Since all mappings ω_i have the same structure, only the offsets d_i^{x} and d_i^{y} are shown. The upper plot shows the graph $F_{\mathcal{P}}$ associated with the program. The lower plot shows the $IFS_{\mathcal{P}}$ using the rectangular notion as introduced in Section 3.4 and the attractor computed using Algorithm 3.45. Note that the size of the area shown is only $R_{\mathsf{m}} \times R_{\mathsf{m}} = 0.33 \times 0.33$ (not 1×1).

For the first and second example the correctness of the translation was shown in Section 5.2 and 5.3. The correctness of the third example is not shown yet, but the visual inspection supports the hypophysis that the algorithm is applicable for a larger class of programs.

Natural Numbers

$\mathcal{P}$:

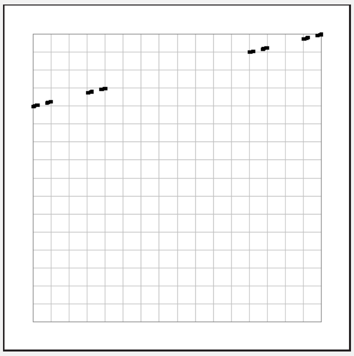
```
n(0).
n(s(X)) :- n(X).
```

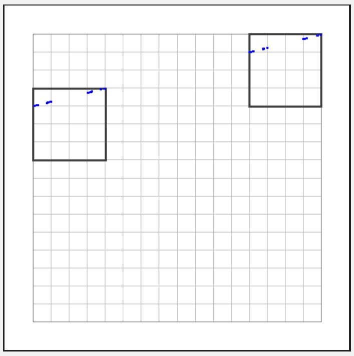
$|\cdot|$:

$|n(s^x(0))| = x + 1$

offsets of the $IFS_{\mathcal{P}}$ for $P = 1$:

$\begin{pmatrix} 0 \\ 0.19 \end{pmatrix}, \begin{pmatrix} 0.25 \\ 0.25 \end{pmatrix}$





Even and Odd Numbers

$\mathcal{P}$:

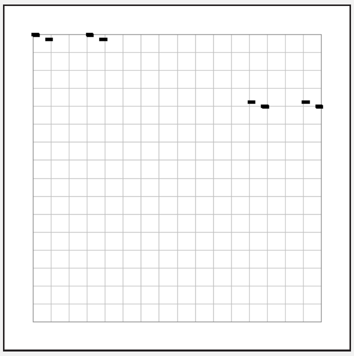
```
e(0).
e(s(X)) :- not e(X).
o(X) :- not e(X).
```

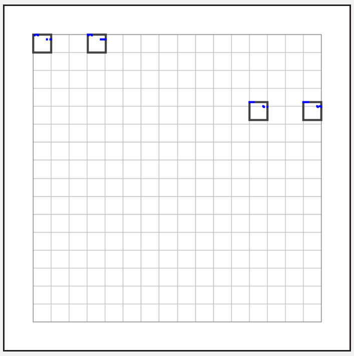
$|\cdot|$:

$|e(s^x(0))| = 2 \cdot x + 1$
$|o(s^x(0))| = 2 \cdot x + 1$

offsets of the $IFS_\mathcal{P}$ for $P = 2$:

$\begin{pmatrix} 0.0 \\ 0.31 \end{pmatrix}, \begin{pmatrix} 0.06 \\ 0.31 \end{pmatrix}, \begin{pmatrix} 0.25 \\ 0.23 \end{pmatrix}, \begin{pmatrix} 0.31 \\ 0.23 \end{pmatrix}$





Seen (since the plot looks a little like the arabic letter "Seen")

$\mathcal{P}$:

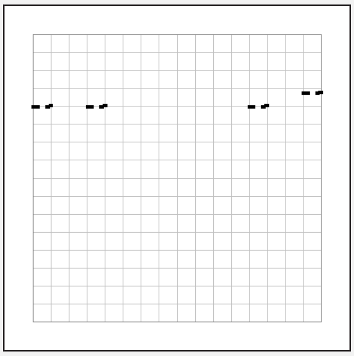
```
p(0).
p(s(X)) :- p(X), r(X).
```

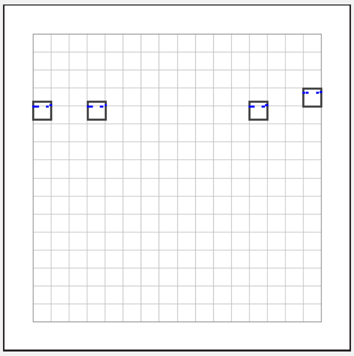
$|\cdot|$:

$|p(s^x(0))| = 2 \cdot x + 1$
$|r(s^x(0))| = 2 \cdot x + 1$

offsets of the $IFS_\mathcal{P}$ for $P = 2$:

$\begin{pmatrix} 0.0 \\ 0.23 \end{pmatrix}, \begin{pmatrix} 0.06 \\ 0.23 \end{pmatrix}, \begin{pmatrix} 0.25 \\ 0.23 \end{pmatrix}, \begin{pmatrix} 0.31 \\ 0.25 \end{pmatrix}$





For the following example Algorithm 5.10 obviously produces a wrong result, at least for periodicity $P = 2$. Unfortunately, the fractal interpolation does not work either, since the program is not acyclic. Still the plot shows some pattern that could indicate self-similarity.

Nice but Wrong

$\mathcal{P}$:

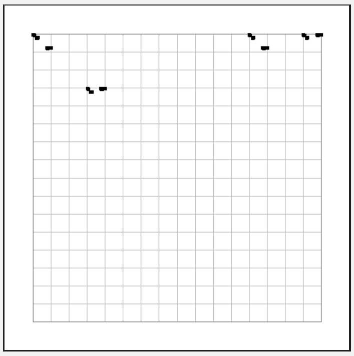
```
p(0).
p(s(X)) :- p(X).
p(X) :- not p(X).
```

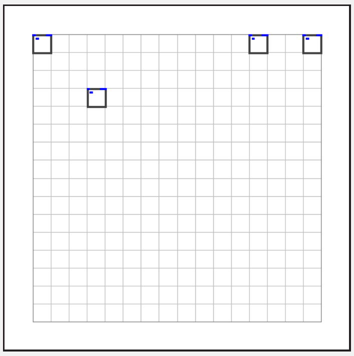
$|\cdot|$:

$|p(s^x(0))| = x + 1$

offsets of the $IFS_{\mathcal{P}}$ for $P = 2$:

$\begin{pmatrix} 0.0 \\ 0.31 \end{pmatrix}, \begin{pmatrix} 0.06 \\ 0.25 \end{pmatrix}, \begin{pmatrix} 0.25 \\ 0.31 \end{pmatrix}, \begin{pmatrix} 0.31 \\ 0.31 \end{pmatrix}$





C. Software

This Section gives a short introduction into the program “LP2IFS” written by the author and used to create the plots of logic programs and their corresponding iterated function systems. The program is written in Java using version 1.4 (on SuSE Linux 8.1). All sources are available at http://www.louise15.de/LP2IFS. Additionally, an applet is made available on the web-site, which allows to create own plots.

The user interface contains two main tabs, namely “LP2IFS” and “Graphs”. The first is split into six parts, which are shown in Figure 21 and described below. Tab “Graphs” contains the graphs created, namely $F_{\mathcal{P}}$, $IFS^{l}_{\mathcal{P}}$ and $A(IFS^{l}_{\mathcal{P}})$.

Parsing the program: Figure 19 shows the BNF of the Prolog parser. This is a restricted version of the usual prolog syntax. There are no lists, cuts or arithmetic expressions allowed and the usual built-in predicates like “is” are not supported. The corresponding scanner produces <name>-, <variable>- and <number>-tokens, which denote function or predicate symbols, variable names and numbers. They are detected according to the usual Prolog syntax, e.g. “p” could be a predicate symbol, “H*_1” could not. A detailed description of the full Prolog syntax can be found in [Apt97].

```
<PROGRAM> ::= <CLAUSES>
<CLAUSES> ::= <CLAUSES> <CLAUSE> | <CLAUSE>
<CLAUSE>  ::= <ATOM>"." | <ATOM> ":-" <BODY>"."
<ATOM>    ::= <name> | <name>"("<TERMS>")"
<LITERAL> ::= <ATOM> | "not" <ATOM>
<BODY>    ::= <LITERAL> | <BODY>"," <LITERAL>
<TERMS>   ::= <TERM> | <TERMS>"," <TERM>
<TERM>    ::= <variable> | <name> | <number> | <name>"("<TERMS>")"
```

Figure 19: LP2IFS - Grammar used for the Prolog parser

Computing the ground program: After entering a program and pushing the “parse”-button the program is parsed. Furthermore, the Herbrand universe $\mathcal{U}_{\mathcal{P}}$ and the predicate symbols $\mathcal{R}/n$ are updated, which are shown in the lists Up: and R/n: respectively. The ground program (ground(P):) is computed by adding every possible ground instance of the clauses, with respect to the Herbrand universe.

Herbrand base and level mapping: After the ground program is computed every atom contained is added to the Herbrand base. Furthermore, every possible atom with respect to the Herbrand universe and the predicate symbols is added. All atoms of the Herbrand base together with their corresponding level are shown in the area |x|:. Within this area the type of level mapping can be chosen. There are several built-in types: “random”, “length”, “reverse length”, “lexical”, “reverse lexical” available. The levels are computed by ordering the atoms randomly, according to their length or their lexical order and taking the position in the sorted list as its level. Furthermore, the levels can be changed directly in the table, which allows every possible level mapping.

The $IFS^l_\mathcal{P}$ for the program: Finally, the $IFS^l_\mathcal{P}$ is constructed according to Algorithm 5.10. The periodicity can be changed using the input-field and pushing the button next to it. The area named "`wrong offset`" contains the interpretation $\frac{T_\mathcal{P}(\emptyset)}{P}$ and the corresponding real value $\frac{f_\mathcal{P}(0)}{B^P}$. The table below shows the interpretation `Dx`, associated with every d_i^x as mentioned in Remark 5.8 and their immediate consequences `Tp(Dx)`. In addition, the real values d_i^x and $d_i^\mathsf{y} = f_\mathcal{P}(d_i^\mathsf{x}) - \frac{f_\mathcal{P}(0)}{B^P}$ are shown.

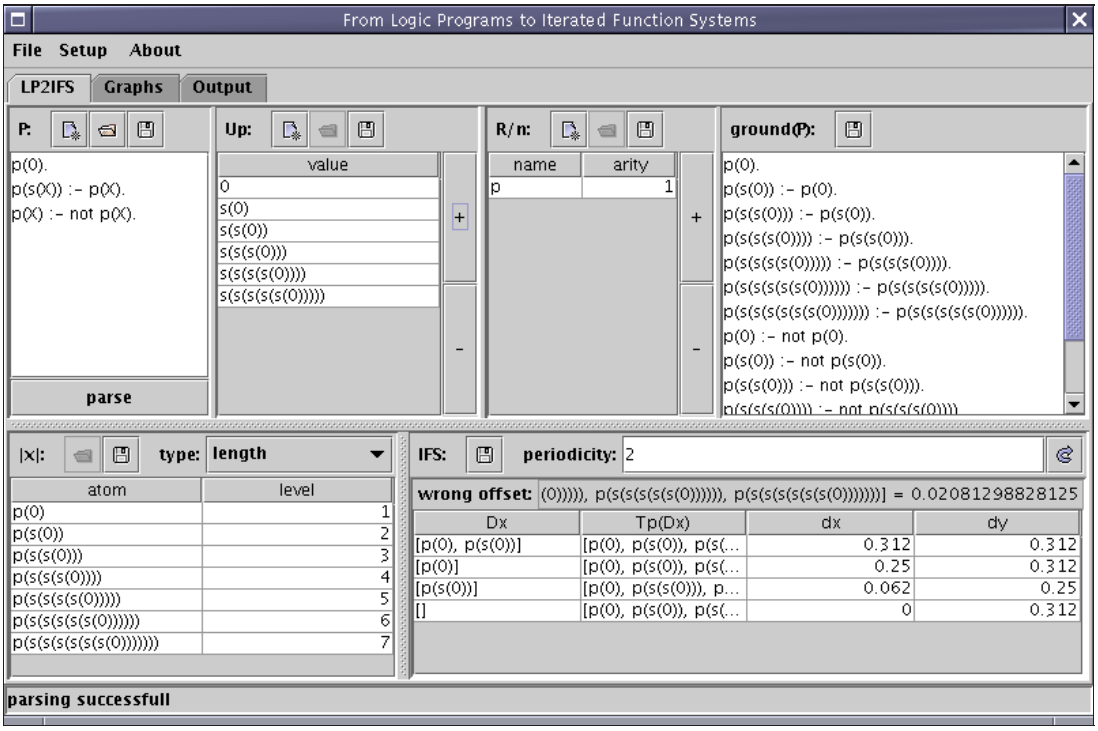


Figure 20: LP2IFS - Setup of a new transformation

`P:`	Prolog program (restricted to the grammar described above). The "parse"-button checks the program and re-computes the corresponding ground version and Herbrand base.
`Up:`	The Herbrand universe, where terms can be added and removed using the buttons on the right side.
`R/n:`	The predicate symbols.
`ground(P):`	The ground version of the program computed using the Herbrand universe.
`|x|:`	The Herbrand base and the level mapping
`IFS:`	The $IFS^l_\mathcal{P}$ constructed as given by Algorithm 5.10.

Figure 21: LP2IFS - Areas of the program's user interface

Dynamics of the program: Every change influences all depending parts directly, i.e. changing the program updates the Herbrand universe, the ground program, the level mapping and the $IFS^l_{\mathcal{P}}$, whereas a change to the level mapping updates the $IFS^l_{\mathcal{P}}$ only.

Further setup: The base B of the embedding R can be changed using the "`Setup`"-menu. This menu also allows to change the size of the output images. The menu "`File`" allows, for example, to save a whole session, i.e. the program, the level mapping, the $IFS^l_{\mathcal{P}}$ and different combinations of the plots are saved at once.

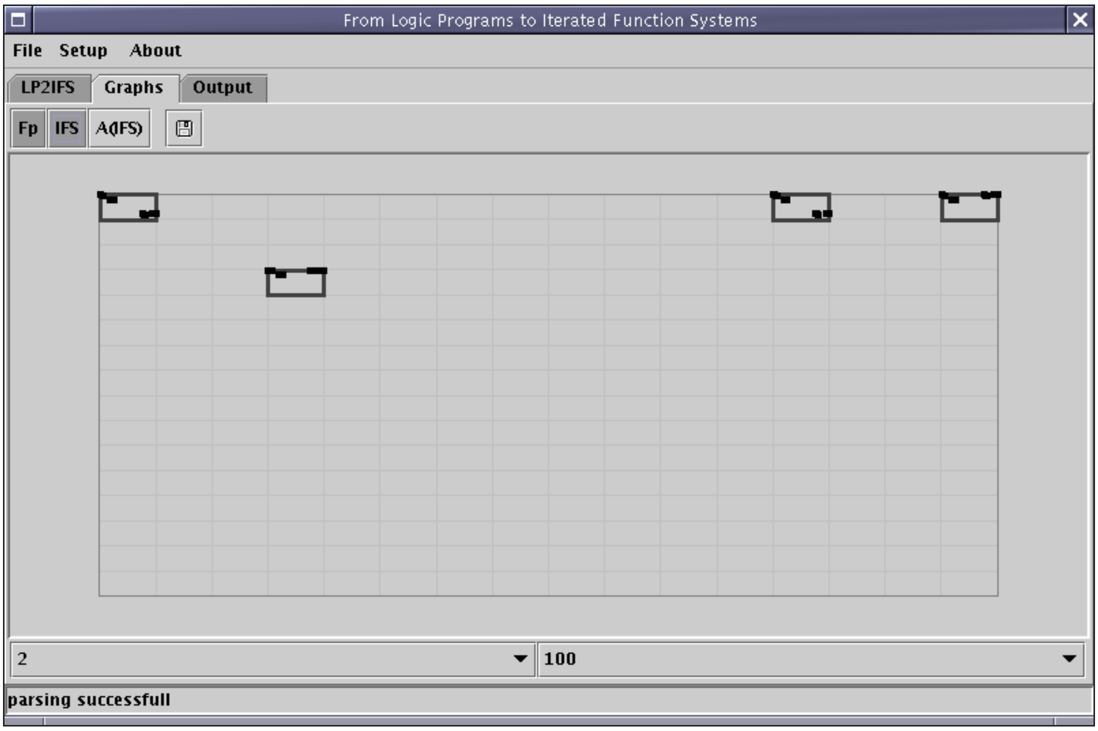


Figure 22: LP2IFS - Graphs produced by the program

Plots created by the program: Figure 22 shows the "`Graph`"-tab of the program. The upper toolbar allows to choose what is plotted, namely `Fp`, `IFS` and `A(IFS)`. These denote the graph of the program, the $IFS^l_{\mathcal{P}}$ and the attractor. Furthermore, it allows to save the current plot as an image. The attractor is computed using the Random Iteration Algorithm (Alg. 3.45). The density of the background-grid and the number of points computed for the attractor can be changed using the lower toolbar.

D. Index of Definitions, List of Figures & References

Index of Definitions

$\dot{\rightarrow}$ (right-shift operator) 30
$\mathcal{B}_{\mathcal{P}}$ (Herbrand base) 5
$\mathcal{C}$ (Cantor set) 19
$d_{\mathcal{P}B}$ (generalised Fitting metric) 32
$D_{\mathsf{f}B}$ (range of R for base B) 26
$F_{\mathcal{P}}$ (graph of $\mathcal{P}$) 31
$f_{\mathcal{P}}$ (embedded $T_{\mathcal{P}}$-operator) 31
h_d (Hausdorff distance) 17
$\mathcal{H}(\mathbf{X})$ 17
IFS (iterated function system) 17
$IFS_{\mathcal{P}}$ 35
$IFS^{l}_{\mathcal{P}}$ (linear $IFS_{\mathcal{P}}$) 39
$IIFS_{\mathcal{P}}$ (interpolating $IFS_{\mathcal{P}}$) 48
R (embedding into $\mathbb{R}$) 26
R_{m} (maximal value of R) 26
$T_{\mathcal{P}}$ (immediate consequence operator) . . . 8
$\mathcal{U}_{\mathcal{P}}$ (Herbrand universe) 5
d_{i} (interpolation metric) 24
Ω^{x}_{n} (set of all mappings in x-direction) . 39
ω^{x} (mappings in x-direction) 39
ω^{y} (mappings in y-direction) 39

acyclic logic program 6
atom 4
attractor 18

bounded 13

Cantor set 19
Cauchy sequence 13
clause 5
closed 13
compact 13
complete 13
continuous 14
contraction 15
convergence 13

equivalent metric spaces 12
equivalent metrics 12
Euclidean Metric 14

fact 5
fractal dimension 19
fractal interpolation function 24
fractal interpolation function for $\mathcal{P}$ 51

generalised Fitting metric 32
graph of the program $\mathcal{P}$ 31
ground 5

Hausdorff distance 17
Herbrand base 5
Herbrand interpretations 7
Herbrand model 7
Herbrand universe 5

immediate consequence operator 8
interpolating iterated function system . . . 48
interpolation data 22
interpolation function 22
interpolation metric 24
isometric spaces 12
iterated function system 17

level mapping 6
limit point 13
linear $IFS_{\mathcal{P}}$ 39
linearity 43
literal 4

metric space 12

program 5

right-shift operator 30

separability 43
space of interpretations 7

term 4
totally disconnected IFS 19

uniformly continuous 14

List of Figures

1. The graph of a real valued $T_{\mathcal{P}}$-operator . . . 1
2. Self-similar structure of the graph . . . 1
3. Five steps of the production of the Cantor set $\mathcal{C}$ . . . 11
4. The first steps of the production of the Sierpinski triangle . . . 17
5. The Sierpinski triangle computed with two different “starting sets”. . . . 19
6. Three times the same fractal? . . . 19
7. Dimensions of different objects . . . 20
8. Illustration of an IFS as a collection of maps . . . 21
9. The similarity of D_{f3} and D_{f4} . . . 28
10. The relation between $T_{\mathcal{P}}$ and $f_{\mathcal{P}}$ . . . 31
11. $T_{\mathcal{P}}(\mathsf{w}_i^{\mathsf{x}}(I)) = \mathsf{w}_i^{\mathsf{y}}(T_{\mathcal{P}}(I))$ holds for the even and odd numbers . . . 47
12. Relations between interpretations, their consequences and their real values 55
13. Interesting regions in the graphs of logic programs . . . 55
14. Two examples of radial basis functions and a RBF-network . . . 56
15. The embedded $T_{\mathcal{P}}$-operator from Example 5.23 . . . 57
16. Dynamics and scheme of an RBF_x^y-unit . . . 57
17. Dynamics and scheme of an $RBF_{s,x}^{x,y}$-unit . . . 59
18. Recurrent $RBF_{s,x}^{x,y}$-network . . . 60
19. LP2IFS - Grammar used for the Prolog parser . . . 69
20. LP2IFS - Setup of a new transformation . . . 70
21. LP2IFS - Areas of the program’s user interface . . . 70
22. LP2IFS - Graphs produced by the program . . . 71

References

[AB94] K. R. Apt and R. N. Bol. Logic programming and negation: A survey. *Journal of Logic Programming*, 19/20:9–71, 1994.

[ABW88] K. R. Apt, H. A. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann, Los Altos, CA, 1988.

[AP93] K. R. Apt and D. Pedreschi. Reasoning about termination of pure prolog programs. *Information and Computation*, 106(1):109–157, 1993.

[AP96] J. J. Alferes and L. M. Pereira. *Reasonning with Logic Programming*, volume 1111 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag Berlin, 1996.

[Apt97] K. R. Apt. *From Logic Programming to Prolog.* Prentice Hall, 1997.

[Bar93] M. Barnsley. *Fractals Everywhere.* Academic Press, San Diego, CA, USA, 1993.

[Bez93] Marc Bezem. Strong termination of logic programs. *Journal of Logic Programming*, 15(1-2):79–97, 1993.

[BF91] N. Bidiot and C. Froidevaux. Negation by default and unstratifiable logic programs. *Theroretical Computer Science*, 78:85–112, 1991.

[Bis95] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[Bry85] V. Bryant. *Metric spaces - Iterations and Applications.* Cambridge University Press, 1985.

[Cav91] L. Cavedon. Acyclic logic programs and the completeness of sldnf-resolution. *Theoretical Computer Science*, 86, 1991.

[CL00] R. Cori and D. Lascal. *Mathematical Logic.* Oxford University Press, 2000.

[Cla78] K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[DM90] C. Darken and J. Moody. Fast adaptive k-means clustering: Some empirical results. In *IJCNN International Joint Conference on Neural Networks*, volume 2, pages 233 – 238, 1990.

[Dub95] S. Dube. Undecidable problems in fractal geometry. *Complexity International*, 2, 1995.

[ESW97] Abbas Edalat, David Sharp, and Lyndon While. Bounding the attractor of an IFS. *Information Processing Letters*, 64(4):197–202, 1997.

[Fit85] M. Fitting. A Kripke-Kleene semantics for general logic programs. *Journal of Logic Programming*, 2:295–312, 1985.

[**Fit94**] M. Fitting. Metric methods three examples and a theorem. *Journal of Logic Programming*, 21(3):113–127, 1994.

[**Fit02**] Melvin Fitting. Fixpoint semantics for logic programming a survey. *Theoretical Computer Science*, 278(1-2):25–51, 2002.

[**Fri98**] B. Fritzke. *Vektorbasierte Neuronale Netze.* PhD thesis, 1998.

[**Fun89**] K. Funahashi. On the approximate realisation of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.

[**GL88**] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In Kowalski and Bowen, editors, *Proceedings of the Fifth International Conference on Logic Programming*, pages 1070–1080, Cambridge, Massachusetts, 1988. The MIT Press.

[**Hit01**] P. Hitzler. *Generalized Metrics and Topology in Logic Programming Semantics.* PhD thesis, Department of Mathematics, National University of Ireland, University College Cork, 2001.

[**HK94**] S. Hölldobler and Y. Kalinke. Towards a new massively parallel computational model for logic programming. In *Proceedings of the ECAI'94 Workshop on Combining Symbolic and Connectionist Processing*, 1994.

[**HKS99**] S. Hölldobler, Y. Kalinke, and H.-P. Störr. Approximating the semantics of logic programs by recurrent neural networks. *Applied Intelligence*, 11:45–58, 1999.

[**Höl01**] S. Hölldobler. *Logik und Logikprogrammierung.* Synchron Verlag, Heidelberg, 2001.

[**HS00**] P. Hitzler and A. K. Seda. A note on relationships between logic programs and neural networks. In Paul Gibson and David Sinclair, editors, *Proceedings of the Fourth Irish Workshop on Formal Methods, IWFM'00*, Electronic Workshops in Comupting (eWiC). British Computer Society, 2000.

[**HS01**] P. Hitzler and A. K. Seda. Continuity of semantic operators in logic programming and their approximation by artificial neural networks. Technical Report WV–01–07, Knowledge Representation and Reasoning Group, Department of Computer Science, Dresden University of Technology, 2001.

[**HSW89**] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[**HW02**] P. Hitzler and M. Wendt. *The Well-Founded Semantics is a Stratified Fitting Semantics*, volume 2479 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag Berlin, 2002.

[**JS93**] J.-S. Roger Jang and C.-T. Sun. Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, 4(1):156–159, Jan 1993.

[Kön02] K. Königsberger. *Analysis 1 and 2.* Springer-Verlag Berlin, 2002.

[Kow79] R. A. Kowalski. *Logic for Problem Solving.* North-Holland, 1979.

[Llo87] J. W. Lloyd. *Foundations of Logic Programming.* Springer Verlag, 1987.

[LMPS95] V. Lifschitz, N. McCain, T. C. Przymusinski, and R. F. Stärk. *Loop checking and the well-founded semantics*, volume 928 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, 1995.

[McI00] Y. McIntyre. *Modellgenerierung mit konnektionistischen Systemen.* PhD thesis, Fakultät Informatik, TU-Dresden, 2000.

[PG90] T. Poggio and F. Girosi. Networks for approximation and learning. volume 78 of *Proceedings of the IEEE*, pages 1481 – 1497, Sep 1990.

[PP90] H. Przymusinska and T. Przymusinski. Semantic issues in deductive databases and logic programs. In R. B. Banerji, editor, *Formal Techniques in Artificial Intelligence: A sourcebook*, pages 321–367. North-Holland, New York, 1990.

[Prz89] T. C. Przymusinski. Every logic program has a natural stratification and an iterated least fixed point model. In *Proceedings of the 8th Symposium on Principles of Database Systems*, pages 11–21, 1989.

[Prz94] T. Przymusinski. Well-founded and stationary models of logic programs. *Annals of Mathematics and Artificial Intelligence*, 12:141–187, 1994.

[PS91] J. Park and I. Sandberg. Universal approximation using radial-basis-function networks. In *Neural Computation*, volume 3, pages 246 – 257, 1991.

[SB90] T. Shepherd and D. Broomhead. Nonlinear signal processing using rbf. In *Proceedings of SPIE*, volume 1348, pages 51 – 61. Int. Soc. for Optical Engineering, 1990.

[Sed95] A. Seda. Topology and the semantics of logic programs. *Fundamenta Informaticae*, 24(4):359–386, 1995.

[SH99] A. K. Seda and P. Hitzler. Topology and iterates in computational logic. In *Proceedings of the 12th Summer Conference on Topology and its Applications: Special Session on Topology in Computer Science, Ontario, August 1997*, volume 22 of *Topology Proceedings*, pages 427–469, 1999.

[vEK76] M. H. van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *ACM*, 23(4):733–742, 1976.

[vGRS91] A. van Gelder, K. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

Contents of the CD

Thesis This thesis as pdf- and ps-file.

Poster A ps-file of the poster presented at the IK2003 in Günne, Germany. For further information on this conference see `http://www.ik2003.de`.

LP2IFS The program used to plot the graphs investigated in this thesis.

Paper A collection of papers concerning logic programming semantics, iterated function systems and connectionist systems. All available papers from the reference list are included.

Software Some additional programs:

- The program "Glito" used to plot the attractors of the iterated function systems.
- JDK1.4 for Windows and Linux

Statement of Academic Honesty

Hereby I declare that this is my work and that I did not use any other source than the ones cited.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass die vorliegende Arbeit von mir selbst verfasst wurde, und ich keine anderen als die angegebenen Quellen benutzt habe.

Sebastian Bader, 2509568
April 30, 2003