Constructive Sets with Non-binary Inclusion

Mikael Fors
Constructive Sets with Non-binary Inclusion

Mikael Fors

Abstract

In this thesis we outline how to represent rough and fuzzy sets in intuitionistic type theory. The inherent meaning of such constructs are valuable when it comes to representing certainty in complex data structures to be used and reasoned on within the context of computer science. We ask fundamental questions regarding how to represent and understand non-binary inclusion within the constructive framework outlined by Martin-Löf and further illustrate the implied limitations and problems associated with the required high level of explicity. While it is possible to simulate non-countable collections, we concern ourselves with sets that are bijective to $\mathbb{N}$ as to showcase the design principle of representation; extendable to more complex collections.
Acknowledgements

I would like to thank my fantastic supervisor, Vera Koponen, who has provided me with great support and given thoughtful input throughout my work with this thesis. Further, I thank the Departments of Mathematics at both Uppsala and Stockholm University for providing such excellent courses and always being reasonable when it comes to administrative tasks.
## Contents

1 Introduction .......................................................... 1

2 Non-crisp sets .......................................................... 3
   2.1 Rough Sets ......................................................... 3
   2.2 Fuzzy Sets ......................................................... 6
      2.2.1 Membership .................................................. 7
      2.2.2 Working with Fuzzy Sets ................................. 7

3 Sets in intuitionistic type theory .................................... 13
   3.1 Basic set definition .............................................. 13
   3.2 Non-canonical Sets .............................................. 17
   3.3 Function Objects and Function Sets ........................... 18

4 Non-crisp Sets in ITT .................................................. 23
   4.1 Simulating Rough Sets .......................................... 23
      4.1.1 Fundamentals ............................................... 23
      4.1.2 The rough set ............................................. 25
   4.2 Simulating Fuzzy Sets .......................................... 30

5 Concluding Remarks .................................................. 34
1 Introduction

In this thesis we consider the task of representing sets having non-binary inclusion, with the possibility of multiple meta-levels of uncertainty regarding said inclusion, within intuitionistic type theory. Such sets are of great utility when considering direct application within computer science as a means of more accurately representing a knowledge image of an underlying binary outline or simply representing subjectiveness. We consider two types of constructs and their associated potential difficulties in terms of representation within a constructive framework. That is, one may view that which is an uncertain collection of knowledge of an existing certainty as well as that which is purely a representation of that which is uncertain. These two types of constructs are separable in their form and meaning, but more importantly so in their inherent being. Said being is an oxymoron as the very idea of a constructive – and thus deterministic – set having a probabilistic flavor seems to defy common sense. However, as we shall see, in all practicality we are still working very much deterministically in terms of our construction while considering certainty rather than probability in terms of meaning.

Intuitionistic type theory is of great interest since it can be utilized in a very direct manner and thus the task of generating a structure upon which reasoning can be done is explicit. With computers becoming increasingly capable of solving non-trivial problems, we must ask ourselves how to deal with more complex information structures typically associated with said non-trivial problem spaces. While the very definition of computation tends to be strongly associated with explicitness, the same does not hold for reality in large. Data tend to be incomplete or uncertain. Is it feasible to expect an absolutely certain premise whenever a computation is to be performed? After all, is this not tangent to the very leap of faith that is an axiomatic system? Indeed, when we assume that something is truly known we disregard reality where there are always uncertainties in terms of absolute validity. While axioms tend to be non-complex in their definition but complex in their implied meaning, assumed certainty imposes the idea that there is simplicity in both. Further, the inherent truth that we judge an axiom to have typically has an associated meta-level justification. We could say that the symmetry axiom of Peano, \( a + b = b + a \), is valid because it matches our understanding of reality; the order in which a collection is created does not typically alter its cardinality should no further process be introduced.
Hence the utility of disregarding absolute certainty is apparent. Within the scope of this thesis we shall study two types of constructs – rough and fuzzy sets – that allow us to outline uncertainty within the set context. A point should be made that while there is a probabilistic flavor to the meaning of the inclusion studied, it is separate from pure probability. Probability tells us what is likely but not certain whereas the non-binary inclusion considered tells us what is certain in terms of what is known; one is dynamic, the other static. How to properly represent knowledge is more important than ever with the explosion of data generation associated with the IT industry success. With this in mind, there is no surprise that we are mostly concerned with application within computer science; a good match considering the design of intuitionistic type theory.

While the theoretical aspects of non-binary inclusion are interesting, it follows that the real aim of the outline studied is to consider applied scenarios in which knowledge of what is known is needed to fully comprehend the task at hand. That is, when one must be able to fully record the knowledge image of a particular data set to properly work with it. This implies added complexity, as will be made apparent within this thesis, but also new exciting possibilities in terms of evaluation.

This thesis is divided into two parts. In part one we discuss the theoretical outline of non-crisp sets; both rough and fuzzy sets are covered. Further, we consider what is meant by a set in intuitionistic type theory. Having covered the required theory, we approach the main purpose of this work: representing rough and fuzzy sets in intuitionistic type theory. We ask ourselves three major questions:

1. How does one represent non-binary inclusion in intuitionistic type theory?

2. How does one work with said constructs and what does the outcome mean?

3. What are the major problems and limitations of the proposed outline?
2 Non-crisp sets

In this section we introduce and discuss both rough and fuzzy sets.

**Definition 1** (Crisp Set). *By a crisp set we mean a classical set, defined as a collection of definite and separate objects as a whole [8] with absolute certainty. An object is either an element of a crisp set or it is not.*

2.1 Rough Sets

The concept of a *rough set* was introduced by Pawlak [12][13]. The idea is that classification of data in terms of membership of a collection can be thought of as three separate layers:

- That which lies in the set for certain
- That which lies outside the set for certain
- That which does not lie in or outside the set for certain

The very definition of the set is done upon a partitioning of the universe considered. That is, by grouping the universe into equivalence classes based on a set of attributes considered, the inherent meaning of set membership is achieved.

Given a universe of objects \( U \) and a non-empty set of attributes \( A \), we construct the information system \( \mathcal{A} := (U, A) \). By an attribute \( a \in A \) we mean a function \( a : U \to K_a \), where \( K_a \) is a set of values that can be associated with said attribute. For instance, the attribute *human height* could take on rational values in a restricted range (measured in centimeters), e.g. \( a(David) = 185 \).

We say that two objects \( u, v \in U \) are *indiscernible* iff they have the same values for all attributes that are related to the set definition we are considering. That is, while the information system might hold a large number of attribute values, many of them may not apply when working with some explicit set. Therefore it is not uncommon to only consider a viable subset of the total attribute set in an applied scenario. For instance, the attribute considered above, which regarded human height, would probably not be of
interest if we were to create a rough set that contains all smart people. We define

\[
\text{IND}(A^*) = \text{IND}(A^* \subseteq A) = \{(u, v) \in U^2 \mid \forall a \in A^* : a(u) = a(v)\}
\]

where \(A^*\) is the subset of attributes \(A\) that we are interested in. It is apparent that \(\text{IND}(A^*)\) is an equivalence relation, meaning we can partition \(U\) into a family of equivalence classes \(U/\text{IND}(A^*)\). Further, by definition \(\text{IND}(A^*) \subseteq U^2\).

It follows from definition that if the cardinality of any of the equivalence classes \([u]_{A^*}\) is greater than one, it becomes impossible to distinguish said elements from each other based on the attributes considered. This is a useful property in an applied scenario where grouping reduces the problem space complexity in that the number of elements that need to be classified can be lowered depending on partitioning attributes.

Let \(X \subseteq U\) be an underlying set that we wish to represent using equivalence classes generated by some attribute subset \(A^*\). Due to the partitioning, it seems probable that there is at least some equivalence class \([u]_{A^*}\) such that it both contains elements in \(X\) and elements in \(\overline{X}\). This is the very premise of the utility of the outline. Namely, we are given a collection of objects that cannot perfectly match some desired class; the ideal explicitness does not match the imperfect reality.

Towards this end, the idea of a rough set can be introduced. The key point is to approximate \(X\) using the available equivalence classes. As previously mentioned, we consider three separate layers, e.g. that which lies in the set with certainty. We define two boundaries: the lower and upper \(A^*\) boundaries. Recall that \(U/\text{IND}(A^*)\) is the family of all equivalence classes with regards to \(\text{IND}(A^*)\), meaning that \(E \in U/\text{IND}(A^*)\) implies that \(E\) is an equivalence class under partition by \(\text{IND}(A^*)\). From this we can define the lower boundary as

\[
A^*X := \bigcup\{E \in U/\text{IND}(A^*) \mid E \subseteq X\}
\]

and the upper

\[
\overline{A^*X} := \bigcup\{E \in U/\text{IND}(A^*) \mid E \cap X \neq \emptyset\}
\]
The area given by $\text{BOUND}_{A^*}(X) := \overline{A^*}X - \underline{A^*}X$ is known as the $A^*$-boundary of $X$. Clearly, if it is empty, then $X$ is crisp with regard to $A^*$, otherwise it is rough. The rough set is typically denoted as a tuple consisting of the lower and upper bound. By dividing the cardinality of the lower bound with the cardinality of the upper bound we obtain a ratio known as the rough set’s accuracy $\alpha_{A^*}(X)$. We can visualize these concepts easily as

![Diagram showing inner and outer bounds of a set X](image)

The set $X$ (grey blob) is captured by the lower and upper bounds

### 2.1.0.1 Membership

Given an equivalence class $[u]_{A^*}$, we can determine the associated level of membership. Equivalence classes not contained in $\overline{A^*}X$ should have a membership degree of 0 and equivalence classes fully contained in $\underline{A^*}X$ should have a membership degree of 1. Further, equivalence classes that lie between the two bounds have a membership degree that lies in $(0, 1)$. With this in mind, it is apparent that the membership degree – as defined below – will always be rational due to the fact that it is a measure of coverage ratio. We write

$$
\epsilon_{A^*}^X(E, X) := \frac{|E \cap X|}{|E|} \in \mathbb{Q}
$$
2.2 Fuzzy Sets

A *fuzzy set* is an extension of a crisp set such that truth, viz. membership, is partial [17]. That is, given a domain $X$, we implicitly define a fuzzy set $A$ through its membership function

$$
\epsilon_A : X \rightarrow [0, 1]
$$

Since $\epsilon_A$ is defined for the entire domain, it follows that for any $x \in X$ the value $\epsilon_A(x)$ denotes the certainty that $x$ belongs to $A$. It is apparent that crisp sets are a special case of fuzzy sets since we can define $\epsilon_A(x)$ to be 1 for all $x \in X_0 \subseteq X$ and 0 otherwise, ensuring binary inclusion.

Typically one differentiates fuzzy sets based on the type of their domains. A discrete domain allows us to define fuzzy sets as elements of $\mathbb{R}^{|X|}$, where each index (of the vector in question) is given by

$$
\frac{\epsilon_A(x_i)}{x_i} \in \mathbb{R}
$$

Note that depending on the nature of the range of the membership function, the vector can lie in e.g. $\mathbb{N}^{|X|}$ or other sets contained in $\mathbb{R}^{|X|}$. This follows trivially from the fact that crisp sets are a special case of fuzzy sets and should we pick say $\mathbb{N}_2$, it is obvious that $\forall x \in X : \epsilon_A(x) \in \{0, 1\}$ which implies $\epsilon_A(x)/x \in \{0, 1\}$.

Apart from using a vector notation, one can rely on sum notation\(^2\) which is more compact and explicit due to limits, i.e.

$$
A = \frac{\epsilon_A(x_0)}{x_0} + \cdots + \frac{\epsilon_A(x_n)}{x_n} = \sum_{i=0}^{|X|-1} \frac{\epsilon_A(x_i)}{x_i}
$$

This notation also has the benefit of being extendable when dealing with continuous domains since we can sum over it using an integral\(^3\)

$$
A = \int_X \frac{\epsilon_A(x)}{x}
$$

---

\(^1\)By $\mathbb{R}^{|X|}$ we mean a $|X|$-dimensional real vector, e.g. if $X := \{a, c, q\}$ then $\mathbb{R}^{|X|} = \mathbb{R}^3$.

\(^2\)This is not an algebraic expression but indicates that $A$ is an ordered set of pairs.

\(^3\)This is still a set operation, i.e. $A$ is a set of an infinite number of ordered pairs.
2.2.1 Membership

Clearly the notion of membership is central to the very definition of a fuzzy set; it implicitly defines what the set itself constitutes. Further, it is apparent that the membership function can be designed rather arbitrarily since it is possible to outline one for many kinds of subjective sets, e.g. the set of all numbers that look large. However, there are some rather obvious restrictions on the membership function of a fuzzy set:

- **Range restriction:** For all \( x \in X \) we have \( 0 \leq \epsilon_A(x) \leq 1 \)

- **Uniqueness of membership:** There is only one associated value \( \epsilon_A(x) \) for all \( x \in X \)

The range restriction is due to the meaning of the function, viz. the probability of inclusion. Uniqueness of membership is set-specific, meaning that while \( \epsilon_A(x) \neq \epsilon_B(x) \) can hold for some \( x \in X \) with \( A \neq B \), each set can only assign one membership value to a particular element. This makes a lot of sense when we consider the semantic outline, say \( A \) is the set of all people that Anna finds handsome and \( x \) is Erik, then the outcome that Anna finds Erik handsome while also finding him not so handsome are contradictory and the set no longer has any meaning.

We can extend upon the notion of membership-functions by considering the range as elements of another fuzzy set (e.g. Gottwalds extension \cite{2}\cite{3} of Weidner \cite{15}\cite{16} and Prati \cite{14}). Then the outcome of \( \epsilon_A(x) = (k_0, k_1) \) denotes that \( x \) belongs to \( A \) with degree \( k_0 \) with degree \( k_1 \), viz. if \( k_0 = 0.3 \) and \( k_1 = 0.8 \) then it is 80\% certain that \( x \) lies in \( A \) with 30\% certainty. Obviously this can be further extended by introducing higher-dimensional tuples, but it quickly becomes difficult to imagine a non-theoretical application of such constructs.

2.2.2 Working with Fuzzy Sets

With crisp sets being included among the fuzzy sets, we can expect the “usual” set operations to work somewhat as expected. However, as will be made apparent, most of them have a probabilistic flavor due to how the membership function works.
2.2.2.1 Equality

For instance, we say that two crisp sets $A$ and $B$ are equal iff they share the same elements, viz.

\[ A = B \iff (\forall a \in A : a \in B \land \forall b \in B : b \in A) \]

Can we extend this notion to fuzzy sets? Not really, as membership is variable and thus it means very little when two sets share the same elements. Consider for instance the set of all people that are tall according to Eric and the set of all people that are short according to Eric. Fix a person, say Axel, and notice that if Axel has a high-valued membership of the set of tall people, he must also have a low-valued membership of the set of short people. Yet, the sets are very different and are hardly equivalent. This rather silly example outlines what it means for two fuzzy sets to be equal, namely to share the same domain and have the same membership-value for each element:

\[ A = B \iff ((\text{dom}(A) = \text{dom}(B) = X) \land (\forall x \in X : \epsilon_A(x) = \epsilon_B(x))) \]

2.2.2.2 Subset and Complement

That $A \subset B$ for crisp $A$ and $B$ means that if $a \in A$ then $a \in B$, which is clearly not applicable to fuzzy sets as shown in the discussion on equality. Rather than viewing a fuzzy set as a collection of elements, it is more useful to view it as an area defined by the membership function. Consider the function $\epsilon_A$ which defines the set $A$ of the smart students in the class according to Lisa, depicted below.

![Diagram of $\epsilon_A(x)$ function](image-url)
Say $\epsilon_B$ is the membership-function of $B$, being the set of the smart students in the class according to Maria. For every individual $p \in X$, where $X$ is the domain over which the sets are defined, it follows that if $\epsilon_B(p) \leq \epsilon_A(p)$ then Marias view is a partial picture of the truth in Lisas set. The reason for this is that each element has a given membership-value which in turn defines its probability, so the area generated by $\epsilon_A$ is an estimation of what $A$ constitutes and if $\epsilon_B$ outlines a more restrictive estimation, then Lisa is not incorrect; only less narrow.

Clearly, the opposite does not hold. If Maria thinks Lars is extremely smart but Lisa thinks he is not, then Marias view is not contained within Lisas and cannot be viewed as a partial picture of Lisas truth. The views are thus not compatible.

So, to summarize, that a fuzzy set $A$ is a subset of another fuzzy set $B$ means that the area defined by $\epsilon_B$ is contained in the area defined by $\epsilon_A$, viz. (where we assume $\text{dom}(A) = \text{dom}(B) = X$)

$$A \subset B \iff (\forall x \in X : \epsilon_A(x) \leq \epsilon_B(x))$$

This idea is easily illustrated as

![Diagram]

So, looking at the graph above, when we refer to the area of $\Sigma \epsilon_A$, we simply mean the area underneath the membership-function in the $[0,1] \times X$ plane. With the notion of the area defined by the membership-function in mind, the concept of complement becomes easier to understand. Let $\text{dom}(A) = X$ be discrete and for all $x \in X$, $\epsilon_A(x) \in \mathbb{N}_2$ (i.e. it is a crisp set). Then the complement $\overline{A}$ is simply the collection of elements belonging to $[0,1] \times X$ that do not lie in the area of $\sum \epsilon_A$, which is equivalent to the standard crisp definition. It is apparent that this notion can easily be
extended to general fuzzy sets since the complement – being the area in the
domain not included in $\sum \epsilon_A$ – can always be constructed. For any given
fuzzy set $A$ with a membership-function $\epsilon_A$ we define

$$\bar{A} : \sum 1 - \epsilon_A = \sum \epsilon_T$$

This definition is sound since if an element $x \in X = \text{dom}(A)$ belongs to
$A$ with probability $\epsilon_A(x) = k$ then clearly the probability that it does not
belong to $A$ is exactly $1 - k$, due to $k \in [0, 1]$.

2.2.2.3 Union and Intersection

The intersection of fuzzy sets $A$ and $B$ is a fuzzy set $C$ that should be
viewed as the representation of what truth is shared among $A$ and $B$. So
just like an intersection of crisp sets can be understood as the area shared
between said sets in a Venn diagram, so too can the intersection $A \cap B$ be
understood as the area shared between $\sum \epsilon_A$ and $\sum \epsilon_B$. The operators that
implement this functionality are called $T$-norms and are typically written as

$$\epsilon_{A \cap B} = \top(A, B) : \{[0, 1] \times \{0, 1\} \rightarrow \{0, 1\}$$

That is, in our context a $T$-norm is simply a function that takes two sets
of values such that they lie in $[0, 1]$ and produce a single set with values
that lie in $[0, 1]$. There are many $T$-norms, but they all share the following
properties:

- $\top(A, B) = \top(B, A)$
- $\top(A, \sum 1) = A$
- If $A \subset C$ and $B \subset D$, then $\top(A, B) \subset \top(C, D)$
- $\top(A, \top(B, C)) = \top(\top(A, B), C)$

The most commonly used $T$-norm when it comes to intersection is also
very easy to understand, namely the G"odel $T$-norm. It is trivial to verify the
properties above when considering its definition:

$$\epsilon_{A \cap B} = \top(A, B) : \sum_{i=0}^{\{|X|-1\}} \min\{\epsilon_A(x_i), \epsilon_B(x_i)\}$$
The Gödel $T$-norm will yield a fuzzy set that is pointwise the largest possible; we maintain all shared levels of truth. However, in some applications this is not sufficient. Consider for instance two fuzzy sets $A$ and $B$ that each represent the partial knowledge of some underlying collection. Say we want to combine said knowledge to gain an understanding of what we actually do know with certainty. Then the Gödel $T$-norm will not help us. To this end there are many other variations of $T$-norms that are better suited. This example, for instance, is an extreme scenario in that we would probably use something along the lines of

$$
\tau(A, B) = \sum_{i=0}^{\lvert X \rvert - 1} f(x_i) = \begin{cases} 
\epsilon_B(x_i) & \text{if } \epsilon_A(x_i) = 1 \\
\epsilon_A(x_i) & \text{if } \epsilon_B(x_i) = 1 \\
0 & \text{otherwise}
\end{cases}
$$

Note that the property $\tau(A, \sum 1) = A$ forces us to always pick the membership of the opposite set as a target value. However, this is a non-issue as $\epsilon_C(x_i) > 0$ if we know something for certain. The case where $\epsilon_A(x_i) = 0$ and $\epsilon_B(x_i) = 1$ gives the result 0 and correctly so; both extreme values indicate a certainty which implies a contradiction, so to be safe we must assume 0.

The union of two crisp sets $A$ and $B$ is

$$
C := A \cup B = \{x \mid x \in A \lor x \in B\}
$$

which we extend to fuzzy sets by considering what is meant area-wise regarding their combined membership-functions. That is, if for some $x \in X = \text{dom}(A) = \text{dom}(B)$ we have $\epsilon_A(x) > \epsilon_B(x)$, then clearly the union $C$ of $A$ and $B$ must contain sufficient truth so that $A \subset C$ and $B \subset C$. Recall that $A \subset C$ means that $C$ contains $A$ by the definition for all $x \in X : \epsilon_A(x) \leq \epsilon_C(x)$, so the union of two fuzzy sets must thus be a third fuzzy set such that it covers the maximum membership of any given element. We can represent this notion in several ways using what is known as $T$-conorms (sometimes $S$-norms). The most common such $T$-conorm – the max $T$-conorm – is also the easiest to use, we have

$$
A \cup B = C = \sum_{i=0}^{\lvert X \rvert - 1} \epsilon_C = \sum_{i=0}^{\lvert X \rvert - 1} \max\{\epsilon_A(x_i), \epsilon_B(x_i)\}
$$
Notice the similarity between $T$-norms and $T$-conorms ($\perp$), indeed

$$\perp(A, B) = 1 - \top(A, B)$$

The properties shared among $T$-norms are applicable to $T$-conorms except that $\perp(A, \sum 0) = A$ rather than $\perp(A, \sum 1) = A$. Finally, let us summarize the outline of union and intersection of fuzzy sets by looking at their visual interpretation. In the illustrations below, please note that we use the Gödel $T$-norm for intersection and the min $T$-conorm for union.
3 Sets in intuitionistic type theory

In this section we discuss what it means for something to be a set in intuitionistic type theory and how to work with said set. We expect the reader to have a rather basic understanding of intuitionistic type theory and refer to [7][6][4][5] for an outline.

3.1 Basic set definition

A set in intuitionistic type theory is something more than just a collection of elements. This is due to the context being very explicit; focus is always on computability, which implies countability and reducibility. That is, computations must always be finite in length, type-bound and exact. This means that any expression must not only be defined in principle, but also be well-understood in terms of what it constitutes. In practice we cannot simply define a set implicitly like is possible in regular set theory, e.g. “the set of all primes”. Instead we must describe precisely what such a set is.

Before we move on to the formal definition of a set in intuitionistic type theory, we must make clear the distinction between canonical and non-canonical elements. This is best done through a simple example. Consider Peanos construct [1] of the natural numbers, viz.

\[
\begin{align*}
0 & \in \mathbb{N} \\
S(a) & \in \mathbb{N}
\end{align*}
\]

That is, there is an element 0 that lies in \(\mathbb{N}\) and every element of \(\mathbb{N}\) has a successor that also lies in \(\mathbb{N}\). Clearly, this construct allows us to fully populate \(\mathbb{N}\), as it implies that there is no supremum and further tells us exactly how to form any natural element we can think of. The use of a successor function \(S(n)\) – rather than explicit numbers allow us to construct arbitrarily large numbers with ease, not to mention that it more naturally reflects the very underlying notion of the natural numbers, cf. historical means of counting by carving lines on a stick.

We ask if \(S(S(0))\) lies in \(\mathbb{N}\), and clearly this is the case because it is possible to reduce the expression to the empty string by applying reversed forming rules. Now consider the term \(2^{100^{2^{200}}}\). Does it lie in \(\mathbb{N}\)? By applying some elementary knowledge of mathematics we can deduce that it does,
but the problem is that we cannot state that it belongs to \( \mathbb{N} \) directly as we must
first evaluate the expression. Furthermore, we are taking a shortcut by using deduction. Should we
have evaluated the membership mechanically according to the population rules\(^4\) outlined by
Peano we would have a very time-consuming task ahead of us. In intuitionistic type theory we
distinguish between these two terms because one of them could easily be verified in terms of membership
while the other required further evaluation.

So to summarize, we say that a term or expression is canonical if it refers to something
directly without any intermediary. Similarly, we call a term or expression that refers to something
through some intermediary non-canonical. The difference is thus in terms of the path of how to refer to the
object in question. That is, either we refer to it directly or we refer to it through some other means
such that it still is possible to conclude exactly what we are referring to, but some steps must be
taken to reach said conclusion. As an example, if somebody were to ask me where I go to school and I
say that I attend school at Sveavägen 65 in Stockholm, it would require them to take an extra step (looking
up what school has said address) compared to me just saying that I attend Stockholm School of Economics.

With this distinction in mind, we can move on. To define a set we are required to prescribe how a
canonical element of it is formed as well as how two equal canonical elements of it are formed. By
providing this information we are explicitly defining what is meant when we say that some object is an
element of the set in question, which in turn defines what the set constitutes. We write

\[ A \in \text{set} \]

to denote that \( A \) is a set. This proposition means four things:

1. It is defined when \( a \) is a canonical element of \( A \), written \( a \in \text{el}(A) \)

2. It is defined when two canonical elements \( a \) and \( b \) of \( A \) are definitionally
equal, written \( a = b \in A \)

3. For all \( a \in \text{el}(A) \), canonical reflexivity holds, i.e. \( a = a \in A \)

\(^4\)These are the previously mentioned rules of construction that outline how to form all elements of
the set of natural numbers, i.e. how to populate the set in question.
4. For all \( a, b, c \in \text{el}(A) \), canonical transitivity holds, i.e. if \( a = c \in A \) and \( b = c \in A \), then \( a = b \in A \).

Note that we make a point of using \( \text{el}(A) \) rather than \( A \) when referring to elements. This is because – as already pointed out – a set in this context is something more than just a collection of elements. Therefore we say that \( a \) is an element of the elements of the set \( A \) rather than just an element of \( A \). Points three to four ensure that the equality between canonical elements that is required to be defined during the set formation is an equivalence relation\(^5\). For equality we work directly towards \( A \) and not \( \text{el}(A) \) because equality is supported by the set outline rather than its elements. This will become more apparent in the example below where we create a somewhat elementary set in intuitionistic type theory.

Recall that when we form crisp sets in ordinary set theory we are usually just performing one operation, e.g.

\[ S := \left\{ \frac{a}{b} \in \mathbb{Q} \mid 0 \leq a \land 0 < b \land 2 \mid b \right\} \]

This is radically different from the set formation operation required in intuitionistic type theory; a direct result of the added information required. Let us build an intuitionistic counterpart for \( S \) defined above. First we must assert that \( S \) is indeed a set and we write

\[ S \in \text{set} \]

which we follow by populating the set. In this particular case we are dealing with a set of infinite cardinality, but since it is countable this is not a problem. In practice it will be impossible to express a non-finite element by any means and it is therefore sufficient to lay down population rule that outline what it means to be a member of the set. Note that this entire construct presupposes that \( \mathbb{N} \) is a defined set (again in the flavor of Peano). We write

\[
\begin{align*}
(0, S(S(0))) & \in \text{el}(S) \\
(a, b) & \in \text{el}(S) \\
(S(a), b) & \in \text{el}(S) \\
(a, S(S(b))) & \in \text{el}(S)
\end{align*}
\]

\(^5\)It should be mentioned that it is also the case that symmetry holds, i.e. if \( a = b \in A \) then \( b = a \in A \).
That is, there is an element equivalent to \((0, 2)\) that lies in \(S\) and for any element \((a, b)\) of \(S\), there is an element \((a + 1, b)\) that also lies in \(S\). Further, for any element \((a, b)\) of \(S\), there is an element \((a, b + 2)\) of \(S\). So, to summarize, there is a tuple \((0, 2)\) that represents \(\frac{0}{2}\) that lies in \(S\) and all other elements of \(S\) are greater than this number in addition to all having a denominator that is divisible by two.

Note how a tuple is used to represent \(\mathbb{Q}\)-values, a common strategy since we are always confined to working in a \(\mathbb{N}^n\) space (a direct consequence of the enforced computability). We must then ensure that we properly define what it means for two elements of the set to be equal. This is rather easy since we know that

\[
\frac{a}{b} \times \frac{b}{a} = \frac{ab}{ab} = 1
\]

and if two rationals are equal, say \(a/b = c/d\), then

\[
\frac{a}{b} \times \frac{d}{c} = \frac{a}{b} \times (\frac{c}{d})^{-1} = \frac{a}{b} \times \frac{ab}{a} = \frac{ab}{ab} = 1
\]

so clearly two elements \((a, b), (c, d)\) are equal iff \(ad = bc\). This means we must define multiplication, which in turn requires us to define addition. We do this recursively (here we use ":" rather than "\(\in\)" to denote a non-canonical element):

\[
\begin{align*}
\frac{a}{b} \times \frac{b}{a} = \frac{ab}{ab} = 1
\end{align*}
\]

We are now ready to define equality. In the population rule we implicitly conveyed that each component of an element of \(S\) is a member of \(\mathbb{N}\). However it can be made more explicit by

\[
\frac{(a, b) \in \text{el}(S)}{a \in \text{el}(\mathbb{N})} \quad \frac{(a, b) \in \text{el}(S)}{b \in \text{el}(\mathbb{N})}
\]

We are now ready to define equality as

\[
\frac{a = c = 0 : \mathbb{N}}{(a, b) = (c, d) \in S} \quad \frac{a \times d = b \times c : \mathbb{N}}{(a, b) = (c, d) \in S}
\]
Having done all this, we still need to ensure that the equality we have defined is an equivalence relation.

**Reflexivity**
From the presupposition that \( N \in \text{set} \) we know that \( a \times b = b \times a : N \). Thus, we can let \( c = b \) and \( d = a \) and apply the second equality formation rule above to obtain \((a,b) = (a,b) \in S\).

**Symmetry**
From the assumption \((a,b) = (c,d) \in S\) we know that \( a \times d = d \times a = c \times b = b \times c : N \). Using symmetry again, we can transform \( d \times a = c \times b : N \) into \( c \times b = d \times a : N \). Applying the second rule above on this yields \((c,d) = (a,b) \in S\) and we are done.

**Transitivity**
From the assumptions that \((a,b) = (c,d) \in S\) and \((c,d) = (e,f) \in S\), we know that \( a \times d = b \times c : N \) and \( c \times f = d \times e \in N \). Clearly \( a \times d \times c \times f = b \times c \times d \times e : N \) and thus \( a \times f = b \times e : N \). By applying the second rule on this we get \((a,b) = (e,f) \in S\) and we are done.

We formalize these notions as

\[
\begin{align*}
(a,b) \in \text{el}(S) & \quad \Rightarrow (a,b) = (a,b) \in S \\
(a,b) = (c,d) \in S & \quad \Rightarrow (c,d) = (a,b) \in S \\
(a,b) = (c,d) \in S & \quad \Rightarrow (c,d) = (e,f) \in S \\
(a,b) = (e,f) \in S & \quad \Rightarrow (a,b) = (e,f) \in S
\end{align*}
\]

Finally, after all this work, \( S \) is properly defined. Note that while all of these steps were performed in order, the actual creation of the set occurs at once, with all steps happening simultaneously.

### 3.2 Non-canonical Sets

We have already pointed out the difference between a canonical and non-canonical element in that the later must be evaluated through some intermediary, viz. computed. One interesting aspect of this distinction is that
we may create non-canonical sets with a similar outline. That is, the non-
canonical set is a collection of elements that can be viewed as computations
that will yield canonical elements of other sets.

For instance, the term $S(0) + S(S(0))$ would result in $S(S(S(0)))$ which
is a canonical element of $\mathbb{N}$. So we could write $S(0) + S(S(0)) : \mathbb{N}$, but
perhaps we wish to store all such computational terms in a collection of
their own. This is what we mean when we say that $A$ is a non-canonical
set; all of its canonical elements are themselves but intermediaries for some
other canonical element in some other set. Regular set formation rules apply,
i.e. we must still define how canonical elements of the non-canonical set are
formed and how equal canonical elements are formed. This means that every
non-canonical set is reducible to a canonical set through carrying out all the
computational steps defined by their elements. We write $A \Rightarrow B \in Set$ to
denote this concept, i.e. by carrying out the computation defined by the
elements of $A$ we can obtain a canonical set $B$ which contains exactly the
output of said computations.

3.3 Function Objects and Function Sets

Functions are very important as they provide a means of mapping sets to
each other and allow us to impose the concepts of action and dynamics to
our set-centric outline. In intuitionistic type theory we view the concept of
function as an object which we can apply on input to produce an output.
The idea of function object makes a lot of sense since it is something separate
from the process it defines. Indeed, let us look at a tangent example in the
world of programming. We run CPython in a standard linux terminal and
compare the type of a lambda function (an anonymous function) to that of its
output. Lines with initial >>> are user input and '#' denotes a comment.

```
Python 2.7.3 (default, Sep 26 2012, 21:51:14) on linux2
>>> f = lambda n: n**n # takes a number n and returns n^n
>>> type(f) # check the type of f
<type 'function'>
>>> type(f(3)) # check the type of f(3)
<type 'int'>
```

Initially, we define a lambda function and note how the function itself is
of a different type than its output, exactly like we would expect. That is, the
function is something separate from what it produces and to generate said output we perform an application. Through this action, viz. application, we combine the function with an input to produce an output, similar to how we can imagine utilizing a tool on some raw material. The function is something in itself; it is separate from the action of application. Therefore the concept of function object makes sense, both in theory and in practice, as shown above (the variable \( f \) holds the lambda function, which is an actual object that we can use as we see fit).

The notion of function object is taken a step further in languages such as JavaScript, where functions are not only objects but are indeed central to asynchronous execution. To explain this further, know that traditionally a program is executed in a linear fashion, i.e. we execute the command at line one, then move to line two and so forth. However, this is hardly ideal when working in a dynamic environment. Imagine that our program is a server and can receive a request at any time. If our program is currently executing a request by Bob while Eva sends a request of her own, Eva will need to wait for Bobs’ request to be completed before the server handles hers. This outlines a first-in-first-out scenario which means that waiting time increases a lot with an increase in the number of requests. One common solution to this would be to allocate a pool of available server resources (a thread pool) which can handle multiple request at a time; so in our example we might be able to serve 20 requests simultaneously. However, if there are 21 requests, we are back to the original scenario (albeit with higher capacity). This is where the asynchronous outline makes its entry. Instead of having a pool of resources, we define a loop which will go through each and every request (no matter how many there are) and perform one step in each. So if Bob sends a request for \( 2 + 2 \) and Eva sends a request for \( 2 + 2 \times 5 \), we would perform \( 2 + 2 \) for Bob and then \( 2 \times 5 \) for Eva during the first loop. This means that even with a lot of requests, the time to get a response will not increase so much (since a simple request can be handled within a few loops).

One request step would, in an applied scenario, be the execution of one function call. Let us illustrate this with a function that takes an integer \( n > 1 \) and returns \( n! \) (where \( n! := 2 \times 3 \times \cdots \times n \))
We first define a function `factorial` – note how `var` indicates a variable, implying that the function is in fact a function object – which takes `n` and then loops `n - 1` times, each time multiplying the result with the loop number; yielding `n!`. In our outline, this means that if Bob sends a request for `factorial(n)`, the server would have to perform `n - 1` loops before completing another step in another request (recall that a step is defined as one function call). If we instead exploit the fact that functions are objects, we could redefine our factorial function and write a function `wrapper` which takes the original request and maps it to the new function.

```javascript
var factorial = function(n, result, cb) {
    if (n) {
        factorial(n - 1, result * n, cb);
    } else {
        cb(result);
    }
}

var wrapper = function(n) {
    factorial(n, 1, function(result) {
        return result;
    });
}
```

Now only one multiplication will be done every step since each function call will only require us to calculate the arguments for the next call. Once `n` becomes 0, we call the `callback` (cb) function object with our result. This design depends upon the concept of function object, as without it we would be forced to hard code what to call once `factorial` has computed `n!`. With the solution outlined above, we could change the argument supplied to `factorial`
to be any function we can think of. So for instance, say we instead want to calculate $5n!$ if $n$ is odd and otherwise return $2n!$, we would change `wrapper` to

```javascript
var wrapper = function(n) {
    factorial(n, 1, function(result) {
        return n % 2 ? 2 * result : 5 * result;
    });
}
```

Such dynamics is very appealing and the ability to design a program which passes around function objects allow us to easily separate calculation steps, enabling the server to handle requests as previously discussed. The notion of $f$ and $f(3)$ as being fundamentally different concepts is presupposed within applied computer science, as made apparent in the above examples. Now let us investigate how this notion is reflected within the theory of intuitionistic type theory.

There is a strong similarity between a non-canonical element and a function, yet they are fundamentally different in that a non-canonical element is something that is “in the making” while a function object is a static formula. Consider, for instance, the difference between $f(x) := x + 2$ and $2 + 3$. The first is a function object which can be applied on input and produce an output through some combination of operations that are well-defined while the other is a calculation that is to take place. One is dynamic and the other is not. Further, when dealing with a non-canonical element, we are already in the process of conducting work; it is clear what must be done. A function object requires an act of application before it has any real meaning in terms of what it is to produce. This notion is reflected by us always applying a function object to an input very explicitly. We write `app[f, x]` to produce a non-canonical element of the set that is the range of $f$. It is easy to see why application produces a non-canonical element, just consider the application of the previous example of $f$ on some arbitrary numbers, say 3. We get $3 + 2$, which is clearly a non-canonical element. Note that we write `app(f, x)` when the input is not a canonical element.
Definition 2 (Application Notation). Since \( \text{app}[f, a] \) is a non-canonical element and we sometimes want to show the final result of the computation, it is common to write

\[
el(A) : \text{app}[f, a] \Rightarrow b \in \text{el}(B)
\]

as to convey the output and type. Similarly, it is common to consider a collection of functions. We denote such a collection as a function set and denote it \( A \rightarrow B \in \text{Set} \) to show what type of functions it contains. That is, it contains functions that take a canonical element of \( A \) and produce output that lies in the non-canonical set \( B \), namely the collection of \( \text{app}[f, a] \). Said non-canonical set may then be reduced to a underlying canonical set. Both function objects and sets require us to define computation rules that explicitly state what output is produced. Generally we consider two function objects such that they share domain and produce the same output for the same input as equal, but it is important to note that we are sometimes concerned about the computation itself, viz. \( 2 + 2 = 2 \times 2 \) but they are clearly computationally different.

Finally, function sets are important when we approach the concept of subset within intuitionistic type theory.

Definition 3 (Subset). That \( A \) is a subset of \( B \), written \( A \subseteq B \in \text{Set} \), means that there is a function set \( C := A \rightarrow \{0, 1\} \in \text{Set} \) in which the elements that lie in \( A \) are exactly those that are reduced to 1. That is, a subset presupposes that there is a method of deciding which elements it should contain.
4 Non-crisp Sets in ITT

In this section we begin our work with implementing rough and fuzzy sets within the context defined by intuitionistic type theory. It is apparent from previous discussion that one of the main challenges will be how to represent the sets to ensure that all the details required to properly define them are made possible. This is achieved by first defining constructs that can properly represent the outline in question, followed by ensuring that what we present is indeed a set. Recall that to state that some construct $A$ is a set is equivalent to laying down said proposition in conjunction with population rules for canonical elements as well as a definitional equality.

4.1 Simulating Rough Sets

The rough set is understood as the lower and upper $A^*$ boundaries, i.e. $X := (\overline{A^*}X, A^*X)$. To properly define this construct in intuitionistic type theory there are many steps that need to be taken. First, we must consider how to represent the fundamental parts, e.g. attributes, equivalence classes and elementary objects. Let us begin by considering the fundamental universe in which the rough set is to be defined.

4.1.1 Fundamentals

The basis for any rough set is an information system $\mathcal{A} := (U, A)$. Our first task is to properly define the sets $U$ and $A$. It is apparent that due to context restriction, i.e. sets need to be countable, we can only really consider discrete rough sets.

**Definition 4** (Bijective Sets). Given two sets $A$ and $B$, we say that $A$ is **bijective** to $B$ if and only if there exists a (bijective) function $f$ that provides a one to one mapping between them. That is, all elements $a \in A$ result in a unique element $b \in B$ through the application $f(a)$ and all elements $b \in B$ can be obtained through the input of some $a \in A$ to $f$. Further, the opposite also holds, meaning that the inverse of $f$, denoted $f^{-1}$, will map each element of $b \in B$ to a unique element $a \in A$ such that all elements of $A$ are generated when all $b \in B$ are used as input. We say that two sets are **equinumerous** — they have the same cardinality — if they are bijective.
As \( \mathbb{N} \) is bijective to both \( \mathbb{Q} \) and \( \mathbb{Z} \), it is sufficient to assume that both \( U \) and the range of \( A \) are subsets of \( \mathbb{N} \). We can represent elements of both \( \mathbb{Q} \) and \( \mathbb{Z} \) as elements of \( \mathbb{N}^2 \), so by working with \( \mathbb{N} \) we know that the result can be extended to other countable sets. That this is the case follows from the fact that if \( S_0 \) and \( S_1 \) are bijective, there is a function \( f \) such that it allows us to take any element of \( S_0 \) into a unique element of \( S_1 \) [8]. In our context this means that we can view \( s \) in \( S_0 \) as a non-canonical element of \( S_1 \) implicitly through \( f : S_0 \rightarrow S_1 \).

We begin by redefining what an information system is within our current type-centric context.

**Definition 5 (ITT Information System).** Given a finite canonical set \( U \in \text{Set} \) and a canonical function set \( A := U \rightarrow N_0 \in \text{Set} \) that produces a non-canonical set \( N_0 : \text{Set} \) which can be reduced to a canonical set \( N \) such that it is bijective to some (potentially non-proper) subset of \( \mathbb{N} \in \text{Set} \), we can produce a tuple \( A := (U, A) \in \text{Set} \) which represents an information system in accordance with rough set theory.

With this definition in mind, let us consider some important implications:

- That \( U \in \text{Set} \), means that every conceivable subuniverse comes equipped with a definitional equality “=“ (a fact that also follows trivially from the assumption that \( U \) is countable and bijective to some subset of \( \mathbb{N} \) – we can reuse the equality defined for the natural numbers)

- According to the definition of an attribute set within rough set theory – it is a set of functions \( a : U \rightarrow K_a \) (where \( K_a \) is just the range of possible values for \( a \)) – we understand the information system as something partly indefinite, in that the canonical function set is a calculation waiting to happen. Further, since – by definition – each range is surjective to \( U \) which is assumed finite, it follows that there are some restrictions to the type of the various attributes. While it is the case that the range of any attribute set defines its type, two attributes need not share the same output type, e.g. *age in years* (\( \mathbb{N} \)) and *number of exams passed out of total exams taken* (\( \mathbb{Q} \)), so each attribute range is required to be bijective to a subset of \( \mathbb{N} \). This also means that it is impossible to consider any attribute which requires a precise value in an uncountable set. This is, however, not a real problem in any applied
scenario as illustrated by the fact that no irrational number can ever be properly stored in the memory of any computer that will ever be built. That is, precision is only important up to a certain point, allowing us to simulate sets such as \( \mathbb{R} \) by limiting the precision.

- The definition of the tuple as a set presupposes that we know how to form all elements of the two sets \( A \) and \( U \), meaning that their population rules must be laid down at the moment of set formation when it comes to \( A \). Obviously this also includes defining their respective definitional equalities.

The statement \( A \in Set \) implies that \((U, A) \in Set\). An ordered pair of sets should be viewed in accordance with the notion that \((A, B)\) refers to the construct of \(\{\{A\}, \{A, B\}\}\). That this is a set follows trivially from the fact that \(A\) and \(B\) are presupposed to be sets and clearly a set with a finite number of elements can be formed in a finite number of steps through simple inclusion. Equality is also trivial since it is already well defined for sets through their composition and formation.

4.1.2 The rough set

We now understand the context in which we are working from a type-centric perspective. However, we still need to outline how to create a rough set. It is clear that to do so we need to look at the partitioning process which involves dividing the universe into equivalence classes.

From rough set theory we recall that when forming some explicit rough set, we are concerned with a subset of available attributes. Namely those attributes that “make sense” given the inherent meaning of the set in question. We can consider the output of applying a subset \( A^* \) of \( A \) on \( u \) as a tuple, where \( A^* = \{a^*_0, \ldots, a^*_{n-1}\} \):

\[
  u_{A^*} := (app[a^*_0, u], app[a^*_1, u], \ldots, app[a^*_|A^*|_{n-1}, u])
\]

To create equivalence classes \([u]_{A^*}\) is thus a rather straightforward task, seeing as how the tuple will be understood once the cardinality of \( A^* \) is known. This means that the equivalence classes presuppose that the non-empty and finite subset \( A^* \) of the countable attribute set \( A \) is defined and that the universe \( U \) is both defined and countable.
Subsets in intuitionistic logic are generally understood as constructs that depend on some propositional function. Recall the definition of a subset (definition 3), where the statement that $A$ is a subset of $B$ presupposes that we have a function set with domain $B$ in which all elements of $A$ are mapped to 1 and all elements not in $A$ are mapped to 0. We may consider a family under such a function $P(x)$, denoted

$$P(x) \ [x \in \text{el}(X)] : \text{Set}$$

The content of the brackets are to be understood as a presupposition of the function itself. That is, given that $x$ is a canonical element of $X$, the propositional function $P(x)$ defines a family on $X$ in accordance with an implicit selection process [9]. In other words, if I happen to have a propositional function $P$ that is defined over $X$ – which is presupposed to be a set – then clearly I can form a non-canonical set of function applications of $P$ on all the elements of $X := \{x_0, \cdots, x_n\}$, i.e.

$$\{P(x_0), P(x_1), P(x_2), \cdots, P(x_n)\}$$

Since the propositional function have the range $\{0, 1\}$ it follows that a subset can be formed in accordance with

$$\{x \in \text{el}(X) \mid P(x)\}$$

This is how $A^*$ would be formed in practice, i.e. a selection process would be chosen and then formed into a propositional function that would evaluate each attribute. Since the domain, namely $A$, is presupposed to be a set and the propositional function is presupposed to be defined, we know how to form the elements of $A^*$ and as equality in $A^*$ is implied by equality in $A$, we have everything we need to state that $A^* \in \text{Set}$.

**Definition 6 (Set domain function notation).** We write $\text{app}[F, u]$ to denote the set $\{\text{app}[f_0, u], \text{app}[f_1, u], \cdots, \text{app}[f_{|F|-1}, u]\}$ where $F := \{f_0, \cdots, f_{|F|-1}\}$.

Recall that a tuple $u_{A^*}$ is simply an $|A^*|$-dimensional vector that contains the application of all functions $a_i^*$ that lie in $A^*$ on an explicit element $u$ from the universe $U$. Since a universe $U$ will most likely contain more than one element, it makes sense to consider the set of all such tuples, which we denote $U_{A^*}$ and define in accordance with

\[\{\bot_1, T\}\]

6Obviously we could extend this to have any natural number of truth values, e.g. $\{\bot_1, \frac{1}{2}, T\}$
\[ A^* \subseteq A \in \text{Set}\quad u \in \text{el}(U) \]
\[ \text{app}[A^*, u] : \text{el}(U_{A^*}) \]

\[ A_1^* \subseteq A \in \text{Set}\quad A_0^* = A_1^* \in \text{Set}\quad u \in \text{el}(U) \]
\[ \text{app}[A_0^*, u] = \text{app}[A_1^*, u] : \text{U}_{A^*} \]

\[ A^* \in \text{Set}\quad u_0 = u_1 \in U\quad \text{app}[A^*, u_0] = \text{app}[A^*, u_1] : \text{U}_{A^*} \]
\[ u_{A^*} \in \text{el}(U_{A^*})\quad u_{A^*} = u_{A^*} \in \text{U}_{A^*} \]

That is, the set domain function application of the function set \( A^* \) on an explicit element of the universe yields a non-canonical element of \( U_{A^*} \). Further, we assure ourselves that equal function sets yield equal non-canonical elements and equal elements of \( U \) yield equal non-canonical results after set function application. We also outline reflexivity. Finally, we state that equal non-canonical elements of \( U_{A^*} \) yield equal canonical elements when computed:

\[ a_0 = a_1 : U_{A^*}\quad \text{el}(U_{A^*}) : a_0 \Rightarrow b_0 \in \text{el}(U_{A^*})\quad \text{el}(U_{A^*}) : a_1 \Rightarrow b_1 \in \text{el}(U_{A^*}) \]
\[ b_0 = b_1 \in U_{A^*} \]

The task is now to partition all elements of \( U_{A^*} \) into equivalence classes. We define an equivalence relation \( R_{A^*} \) that is given by \( A^* \) on \( U \):

\[ u \in \text{el}(U)\quad u_{A^*} = v_{A^*} \in U_{A^*}\quad (u, v) \in \text{el}(R_{A^*}) \]

Equality in \( R_{A^*} \) is defined easily by the fact that ‘=’ is an equivalence relation on \( U_{A^*} \):

\[ u_{A^*} = u_{A^*} \in U_{A^*}\quad v_{A^*} = v_{A^*} \in U_{A^*} \]
\[ u_{A^*} = v_{A^*} \in U_{A^*}\quad u_{A^*} = w_{A^*} \in U_{A^*} \]
\[ v_{A^*} = w_{A^*} \in U_{A^*} \]

So for two elements to be equal in \( R_{A^*} \), their components must all be equal in \( U_{A^*} \). From this relation we can form the equivalence classes in accordance with

\[ (u, v) \in \text{el}(R_{A^*})\quad v \in \text{el}(\{u\}_{A^*}) \]

27
The upper and lower boundary sets must be understood to define the boundary which is the rough set. We already know that these two sets presuppose that the set we wish to simulate is a known subset $X$ of the universe $U$. The lower bound is equivalent to the elements of $[u]_{A^*}$ such that they contain elements which all lie in $X$ while the upper bound is the collection of all elements of $[u]_{A^*}$ which share at least one element with $X$. It is apparent that the upper bound is easier to define, so let us start with that.

**Definition 7 (ITT Upper Bound).** The upper bound of $X$ with relation to $A^*$ is a canonical set that contains all equivalence classes such that at least one element belonging to them lies in $X$. So if $x$ is an element of $X$ and is equal to some other element $u$ in $U$ under the relation $R_{A^*}$, it follows that $u_{A^*}$ must lie in the upper bound. We write

$$
X \subseteq U \in \text{Set} \quad x \in \text{el}(X) \quad u_{A^*} = x_{A^*} \in U_{A^*}
$$

Equality can then be handled trivially by the fact that if two elements of the upper bound are equal in Set, they are the equivalence class and are thus also equal in the upper bound. Reflexivity follows from a similar argument; any equivalence class is equal to itself. We write

$$
a, b \in \text{el}(\overline{A^*}X) \quad a_{A^*} = b_{A^*} \in U_{A^*}
$$

**Definition 8 (ITT Lower Bound).** The lower bound of $X$ with relation to $A^*$ is a canonical set that contains all equivalence classes given by $R_{A^*}$, such that any element in $U$ belonging to them will also belong to $X$. We can define the lower bound using the upper bound and simple set operations.

First, since $X \subseteq U \in \text{Set}$ is presupposed, it follows that its complement $U \setminus X$ is also a set. That this is the case follows from the fact that a subset presupposes that there is a propositional membership function which we can simply invert to form the complement. We may then consider the upper bound of the complement with regards to $A^*$, which covers all classes such that they have at least one element in $U$ that does not lie in $X$. This is the very definition of the complement of the lower bound. As such, all that is left to do is form the complement of $\overline{A^*}(U \setminus X)$ and we have the lower bound. We
formalize this into one simple rule which relies on our previous definition of the upper bound

\[ a \in el(U \setminus \overline{A^*}(U \setminus X)) \]
\[ \Rightarrow a \in el(A^*X) \]

Equality is handled exactly as with the upper bound, i.e. by looking at equality in \( U_{\overline{A^*}} \) and ensuring reflexivity

\[ a, b \in el(A^*X) \quad a = b \in U_{\overline{A^*}} \]
\[ a = b \in A^*X \]
\[ a \in el(A^*X) \]
\[ a = a \in A^*X \]

Finally, with both the lower and upper bounds understood, we are ready to define what it means for a set to be rough in intuitionistic type theory. We know from earlier that a rough set is a tuple \( X_{\overline{A^*}} \) consisting of the lower and upper bounds, so all that needs to be made explicit once the two boundaries are formed is which set is which, i.e. which is the lower and which is the upper bound. This is trivial to do with the following formation rule:

\[ X_{\overline{A^*}} \in Set \]
\[ (0, \overline{A^*}X) \in el(X_{\overline{A^*}}) \]
\[ (S(0), \overline{A^*}X) \in el(X_{\overline{A^*}}) \]

Equality can then be enforced by

\[ (a, b), (c, d) \in el(X_{\overline{A^*}}) \quad a = c \in \mathbb{N} \quad b = d \in Set \]
\[ (a, b) = (c, d) \in X_{\overline{A^*}} \]
\[ a \in el(X_{\overline{A^*}}) \]
\[ a = a \in X_{\overline{A^*}} \]
4.2 Simulating Fuzzy Sets

To simulate a fuzzy set we need to consider how to represent membership. Obviously this cannot be done in terms of elements in \( \mathbb{R} \) since it is uncountable and thus not usable within our computable context. Instead we must consider a membership degree to be a rational number, represented as a pair \((a, b)\) with the inherent meaning \(a/b\). This means that we are limited in the range of membership values, but in practice we can utilize large values for \(b\) to simulate sufficient precision for most applications.

**Definition 9** (ITT Membership set). The membership set \(\mathcal{M}\) is a collection of pairs that presupposes a natural number \(m\) such that it defines the number of possible membership degrees. Given \(m\), we have \(|\mathcal{M}| = m + 1\).

Formation of elements of \(\mathcal{M}\) is straightforward since \(\mathcal{M} \subseteq \mathbb{Q}\). Recall that we presuppose a fixed natural number \(m\) (with an associated relation \(\leq\) defined on \(\mathbb{N}\)).

\[
\begin{align*}
    n & \in \text{el}(\mathbb{N}) \quad n \leq m \in \mathbb{N} & (n, m) & \in \mathcal{M} \\
    a & \in \text{el}(\mathcal{M}) \quad a = a \in \mathcal{M} & a = b & \in \mathcal{M} \\
    (a, m), (b, m) & \in \text{el}(\mathcal{M}) \quad a = b & \in \mathbb{N} \\
    (a, m) & = (b, m) \in \mathcal{M}
\end{align*}
\]

**Definition 10** (ITT Fuzzy set). That \(A\) is a fuzzy set presupposes that there is a universe \(U \in \text{Set}\) that is a subset of \(\mathbb{N}\) such that each element of \(U\) has a defined membership in \(A\). Further, an element \(a\) of \(A\) is a \((n+1)\)-dimensional tuple \(a := (x, e_0^A(x), \ldots, e_{n-1}^A(x))\) where \(x \in \text{el}(U)\) and \(e_i^A(x)\) is the result of the computation of \(i\)-level membership. Membership is represented as an element of \(\mathcal{M}\).

That \(a\) has a 0-level membership \(k/m\) in \(A\) means that it belongs to \(A\) with \(k/m\) certainty. If \(a\) has a \(n\)-level membership of \(k/m\) in \(A\), it follows that the \((n - 1)\)-level membership is \(k/m\) certain. The degree \(d\) of a fuzzy set \(A\) is defined to be the highest level membership considered, meaning that an element \(a\) of \(A\) is a \(d + 1\)-dimensional tuple \((d \text{ membership elements and one element of } U)\).

This construct allow us to generate very explicit chains of meta-data regarding membership. Say we measure temperature at various positions in a
room and define a function defining “hot temperature”. Such a function is clearly subjective and as such we would get an output of “hotness” degree. If we consider a 0-degree fuzzy set, each element would be a pair consisting of a point and a “hotness” degree. However, say we know that our measurement device is only accurate in 95% of the cases, then we could build a 1-degree fuzzy set which would have tuple elements consisting of a point, 0-level membership and 1-level membership. The 1-level membership would then simply tell us how certain the 0-level membership is (in this case 95/100).

When we write $A^d_{F,m} \in \mathbf{Set}$ we mean that $A$ is a fuzzy set with degree $d$ having $(m + 1)$ levels of inclusion.

It is apparent that a non-canonical element of $A^d_{F,m}$ is of the form (where each function $f_A^i$ defines $i$-level membership for any element $a$ of $A$)

$$(x, \text{app}[f_A^0, x], \text{app}[f_A^1, x], \cdots, \text{app}[f_A^{n-1}, x])$$

Since we need to show how $A^d_{F,m}$ is populated in terms of canonical elements, it follows that we must first presuppose a canonical function set which holds all $f_A^i$. Typically, the membership function is but a curve that is drawn upon the cartesian plane $[0, 1] \times U$. That is, it is defined by a constructor who lays down what is known in terms of certainty. Such an act must be performed in our context as well, but rather than drawing a function curve, the constructor must define the mapping which defines computation of function object application.

With this in mind, it follows that to define the set $F^d_A$ which contains all the membership functions, we must lay down computation rules for all its elements. This can be done in many ways, but the essential part is that the range of any such function element is a subset of $\mathcal{M}$ and that the domain is exactly the universe $U$ considered.
The construction requires multiple steps, but they should – as usual – be taken simultaneously. For simplicity, let \( k := |U| \) and presuppose that \( x_i \in \text{el}(U) \)

\[
F^d_A := U \rightarrow \mathcal{M} \in \text{Set}
\]

\[
f_0 \in \text{el}(F^d_A)
\]

\[
f_d \in \text{el}(F^d_A)
\]

\[
\text{el}(\mathcal{M}) : \text{app}[f_0, x_0] \Rightarrow (a^0_0, m) \in \text{el}(\mathcal{M})
\]

\[
\vdots
\]

\[
\text{el}(\mathcal{M}) : \text{app}[f_d, x_k] \Rightarrow (a^d_k, m) \in \text{el}(\mathcal{M})
\]

Note that the \( a^i_j \) values have to be explicitly picked by the constructor and that they do not have to be unique, e.g. \( a^3_0 \) could be equal to \( a^3_7 \). Further, note that all elements in the universe are properly mapped by each function. What remains is for us to ensure equality, which is easy to do:

\[
\frac{f \in \text{el}(F^d_A)}{\text{app}[f, x_0] = \text{app}[f, x_1] : \mathcal{M}}
\]

\[
\frac{f = f \in F^d_A}{f = f \in F^d_A}
\]

\[
\frac{g = f \in F^d_A}{g = f \in F^d_A}
\]

\[
\frac{f = g \in F^d_A}{x \in \text{el}(U) \rightarrow \text{app}[f, x] = \text{app}[g, x] : \mathcal{M}}
\]

\[
\frac{f = g \in F^d_A}{x \in \text{el}(U)}
\]

We can now write

\[
\frac{x \in \text{el}(U)}{F^d_A := U \rightarrow \mathcal{M} \in \text{Set}}
\]

\[
\frac{(x, \text{app}[F^d_A, x])}{a \in \text{el}(A^d_{F,m})}
\]

\[
\frac{a = a \in A^d_{F,m}}{b = a \in A^d_{F,m}}
\]

\[
\frac{(a, e^0_A(a), \ldots, e^d_A(a)), (b, e^0_A(b), \ldots, e^d_A(b)) \in \text{el}(A^d_{F,m})}{a = b \in U}
\]

\[
\frac{(a, e^0_A(a), \ldots, e^d_A(a)) = (b, e^0_A(b), \ldots, e^d_A(b)) \in A^d_{F,m}}{a = b \in U}
\]
Let us consider implicit fuzzy set formation through union and intersection. For simplicity we shall restrict ourselves to the most basic t-norm and t-conorm, namely min and max.

**Definition 11 (Fuzzy Union).** The fuzzy union \((A \cup B)_{F,m}^d\) of \(A_{F,m}^d\) and \(B_{F,m}^d\) is a fuzzy set defined in accordance with the max T-conorm. It thus presupposes that there is a function \(\max(a,b)\) which is defined for the domain \(\mathcal{M}\). Since any element \(a\) of \(\mathcal{M}\) has the form \(\frac{a}{m}\) with \(0 \leq n \leq m\) we can easily define this function as (where \(S(n)\) denotes the successor of \(n\))

\[
\max \in \mathcal{M}^2 \rightarrow \mathcal{M}
\]

\[
\max\left(\frac{0}{m}, \frac{0}{m}\right) \Rightarrow \frac{0}{m}
\]

\[
\max\left(\frac{S(a)}{m}, \frac{0}{m}\right) \Rightarrow \frac{S(a)}{m}
\]

\[
\max\left(\frac{0}{m}, \frac{S(b)}{m}\right) \Rightarrow \frac{S(b)}{m}
\]

\[
\max\left(\frac{S(a)}{m}, \frac{S(b)}{m}\right) \Rightarrow S\left(\max\left(\frac{a}{m}, \frac{b}{m}\right)\right)
\]

Since \(m\) is finite and no nominator can exceed it, it follows that the application of max will always be a finite calculation (at most \(m\) steps required). This allows us to construct set forming rules for the fuzzy union, utilizing the max T-conorm defined above as a non-canonical component in the process. We write (for \(d = 1\), higher degrees would work in a similar fashion for each membership level)

\[
x \in \text{el}(U) \quad (x, \frac{a}{m}) \in \text{el}(A_{F,m}^0) \quad (x, \frac{b}{m}) \in \text{el}(B_{F,m}^0) \quad (x, \max(\frac{a}{m}, \frac{b}{m})) : \text{el}((A \cup B)_{F,m}^0)
\]

So in reality, we are defining what it means to be a non-canonical element of the fuzzy union. This makes sense when we consider that the union will always rely on a T-conorm, which is in effect the very function that is required to reduce the non-canonical elements into canonical elements of the union in question.
Equality would be defined as expected, viz. the canonical result of the tuple output should be equal in their respective types. We write

\[
\begin{align*}
  a &= b \in (A \cup B)^0_{F,m} \\
  b &= a \in (A \cup B)^0_{F,m} \\
  a \in \mathcal{el}( (A \cup B)^0_{F,m} ) &\quad a = a \in (A \cup B)^0_{F,m} \\
  (x_0, k_0), (x_1, k_1) \in \mathcal{el}( (A \cup B)^0_{F,m} ) &\quad k_0 = k_1 \in \mathcal{M} \\
  x_0 = x_1 \in U &\quad (x_0, k_0) = (x_1, k_1) \in (A \cup B)^0_{F,m}
\end{align*}
\]

It is easy to see that to define a fuzzy intersection we need only invert the result from the union outline. That is, the base case for min would pick the opposite index compared to max. Equality and formation would otherwise look the same.

5 Concluding Remarks

We have outlined how one can represent both rough and fuzzy discrete sets in intuitionistic type theory using simple structures and rules. While it is apparent that the forming process of said sets compared to their standard versions is quite extensive and requires multiple steps, it can be automated using computers. Since our ambition is to use the framework of intuitionistic type theory as a means to reason and work with well-defined structures in an applied scenario, it follows that the level of exactness required is not a drawback to the method suggested. Rather, it should be viewed as a strong indicator of the utility inherent to the outline. Indeed, when we study some of the function sets that are required to support computation, they remind us – as they should – of a process quite similar to that of regular programming. This is a very important point to make as it is all too easy to disregard the constructs presented in this thesis as overly verbose or simply theoretical when the practices used are common within the world of programming.

There are, of course, some issues that have not been properly addressed due to scope limitations; one being that it might not be trivial to decide upon a proper data structure when working with more complex data. While it is true that all common data structures such as trees, lists, arrays, queues and stacks can be constructed within the context, it may prove very difficult to design well-defined computational and reduction rules for some of them.
In some cases it might even be more useful to use a less optimized structure simply because it is easier to define.

Some may argue that the heavy dependability on computability is a limitation of the entire theory as it forever confines us to countable sets. However, this is purely a theoretical argument as such limitations are already in effect when working in an applied scenario; we cannot store or properly work with actual non-countable transfinite objects. It is possible to simulate the reals using tuples that only provide a limited accuracy when it comes to non-rational members. This is not different from us accepting that floating point datatypes provide a lot of utility in computer science while having a limited accuracy [11].

The fact that the inherent meaning of the constructs considered within this thesis is perfectly reflected in the creation process, allows us to classify them as analogous to complex objects rather than types. In a strict sense – obviously – they are but types, yet they convey something more by their very nature. We can easily spot the similarity between the set outlines required by intuitionistic type theory and the notion of a class in object oriented programming. This link alone is sufficient justification for continuous efforts to be directed towards implementing and utilizing the theory in practical and applied scenarios.

There are many interesting aspects that should be explored further; perhaps the most important being automatization of set definition generation. Within the explicit context considered, viz. rough and fuzzy set simulations, this is particularly important when we expect generic data that requires a large collection of rules. A simple example would be to imagine the work required to properly define all rules for a $d = 30$ fuzzy set union or intersection; there would be a large number of base cases to consider when dealing with the max or min portions of computation. Also, the hybrid solution of fuzzy rough sets should be explored further as it provides even greater utility in many fields, compared to traditional fuzzy or rough sets [10].
References


