Predicate transformers have been shown to model angelic and demonic nondeterminism. However, conventional relational models do not exist to model both kinds of nondeterminism simultaneously. Recent theories have been shown to model both angelic and demonic nondeterminism using binary multirelations which are equivalent to predicate transformers.

In this thesis, we model both kinds of nondeterminism using lifted multirelations, which are relations between power sets. The correspondences between the set of relations, the set of binary multirelations and the set of lifted multirelations is established using Galois connections. With the help of the above investigations, we list the properties of this lifted binary multirelational model to define the semantics of a program. We show how angelic and demonic nondeterminism can be modeled in this relational model. We then list a few operations for the angelic and demonic nondeterminism. We further define semantics of program and specification constructs for extensions of Dijkstra’s guarded command language.
Acknowledgement

I would like to express my sincere gratitude to my advisor, Dr. Austin Melton, for his invaluable guidance throughout my research period. His constant encouragement and ability to explain things in a simple way enabled me to clearly understand the application of mathematical concepts in computer science.

I would like to dedicate this thesis to my family for their love and support.
Contents

1 Introduction 1

2 Background Concepts 3
   2.1 Syntax and Semantics ......................................... 3
   2.2 Nondeterminism ................................................ 5

3 Related Work 10
   3.1 Binary Multirelations ........................................... 10
   3.2 Predicate Transformer Semantics and Nondeterminism .... 14
   3.3 Galois Connections ............................................. 15

4 Lifted Binary Multirelations 17
   4.1 Definition ....................................................... 17
   4.2 Angelic and Demonic Nondeterminism ....................... 20
   4.3 Refinement Ordering .......................................... 20
   4.4 Program Specifications Using Lifted Binary Multirelations 25

5 Conclusion and Future Work 28
Chapter 1

Introduction

Formal methods are helpful in achieving precision in the software development process, and because of the abstraction which may be imposed in the specification phase, nondeterminism may be introduced at the formal specification level. In program development, nondeterminism is a disadvantage. So, special notations and semantics need to be used in formal specification of programming languages to model nondeterminism in program specification. Nondeterminism is of two types, angelic and demonic. Over the years, models such as functions, binary relations and predicate transformers have been used to model programs, but these concepts are limited by the fact that functions model only deterministic programs and binary relations do not easily show the differences in angelic and demonic nondeterminism. Predicate transformers can model both kinds of nondeterminism simultaneously, but they model program behavior backwards, in the sense that they map a set of outputs to a set of inputs. A relational model which is equivalent to predicate transformers and which is called the binary multirelations model has been introduced by I. Rewitzsky[9]. Binary multirelations are relations from states to sets of states and they can naturally model both kinds of nondeterminism. In this thesis, we further develop a new model which is equivalent to the binary multirelational model and thus also to the predicate
transformer model. This new model is called the lifted binary multirelations model. Lifted binary multirelations are relations between power sets; they relate sets of inputs to sets of outputs. This model was introduced in the thesis by Jidesh Soudamini in [12]. We define how program behavior can be modeled in this relational model. In section 4.2, we further define how both angelic and demonic nondeterminism can be specified using these lifted binary multirelations. In section 4.3, we list operations for the angelic and demonic nondeterminism, namely union and intersection of two lifted binary multirelations. In section 4.4, we define a semantics of program and specification constructs for extensions of Dijkstra’s guarded command language.
Chapter 2

Background Concepts

2.1 Syntax and Semantics

Syntax: A programming language consists of a syntactic notation or syntax, which is a possibly infinite set of elements that can be used in a program. The common term used for the syntactic elements is “expressions”. For example, the syntax for an arithmetic expression, given by a BNF-like grammar, is as follows

\[
\begin{align*}
<Exp> & ::= <Number> | <Variable> \\
& | (<Exp>) | <Exp> \\
& | <Exp> [+ | - | * | /] <Exp> \\
& | FunctionAbe(<Exp>, <Exp>).
\end{align*}
\]

The term syntax is used whenever a reference is to be made to the notation of a language. Syntactic issues focus on the notational aspects of the language, disregarding any meaning. The meaning of a language is defined by its semantics, explained in the next section.
**Semantics:** The meaning of a programming language is formalized by its semantics. A formal, mathematical definition of the language semantics is a good approach to determine the meaning of the program constructs. Advantages of this approach are that the formal descriptions are not ambiguous, are concise and they may allow us to write mathematical proofs for the properties of the programs.

There are three main approaches to specify the semantics of programming languages:

1) **Operational Semantics:** This method describes how a program executes on an abstract machine, which has a state and a set of instructions. The machine is defined by specifying how the components of the state are changed by each of the instructions. The semantics of a particular programming language are then defined in terms of the behavior of the abstract machine. The semantic description of a program in a given programming language is a translation from the abstract machine’s execution of the program.

2) **Denotational Semantics:** This method models programs as mathematical functions. Mathematics provides powerful tools for expressing and modeling the effects of syntactic expressions. The mathematical concept of function provides a very natural and convenient way of modeling the behavior of deterministic programs.

3) **Axiomatic Semantics:** Here, the description of a language is provided by associating an axiom with each statement in the language. These axioms stipulate which conditions have to hold (preconditions) before the execution of a statement so that certain conditions will hold (postconditions) after the execution of the statement.
2.2 Nondeterminism

A deterministic program can be modeled by a function or a partial function in that for a given input, it always produces the same output. As an example, a deterministic finite automata can be considered. The automaton illustrated in figure 1, accepts strings over the alphabet \{a,b\} that begin and end with the same symbol.

![DFA diagram](image)

Figure 2.1: DFA accepts strings beginning and ending with same symbol

Nondeterministic programs may offer a choice of outputs for each input. If a program specification is nondeterministic, then it leaves more choice for implementation and paves the way for better optimization. An example is a simple database query without any sort options; it gives an arbitrarily sorted list of records. Though the implementations are deterministic, the results are determined by state components and implementation details that are not visible to the outsider.

Considering finite automata again, the following nondeterministic finite automaton
illustrated in figure 2 accepts the strings over the alphabet \{0,1\} that contain the substring 101 or 11. The choice of outputs (state transitions) available in the start state and the “\(\varepsilon\)” move constitute the nondeterminism in the program.

Let us now consider a finite automata in figure 3 that produces output. The path descriptions in the following figure are of the form I/O, where I = input and O = output. When 01 is the input, it can yield 01 or 10 as output.

A program can be viewed as a contract between two agents, angel and demon, and in accordance with this, there are two types of nondeterminism- angelic and demonic. Let us consider a specification of a program where a machine and human user interact with each other and make choices. Angelic nondeterminism occurs when the choice is made by the angel, the machine in this case, and it is assumed that the angel will choose the best possible outcome in our favor. Demonic nondeterminism occurs when the choice is made by the demon, the external user, and we cannot make an assumption that the choice made by the demon would be in our favor. The angelic

Figure 2.2: Nondeterministic Finite Automata

Let us now consider a finite automata in figure 3 that produces output. The path descriptions in the following figure are of the form I/O, where I = input and O = output. When 01 is the input, it can yield 01 or 10 as output.

A program can be viewed as a contract between two agents, angel and demon, and in accordance with this, there are two types of nondeterminism- angelic and demonic. Let us consider a specification of a program where a machine and human user interact with each other and make choices. Angelic nondeterminism occurs when the choice is made by the angel, the machine in this case, and it is assumed that the angel will choose the best possible outcome in our favor. Demonic nondeterminism occurs when the choice is made by the demon, the external user, and we cannot make an assumption that the choice made by the demon would be in our favor. The angelic
Figure 2.3: NDFA that produces 01 or 10 as output

and demonic choices in general can be shown as follows. Consider the following finite
automata in figure 4. If we want to guarantee(demonic) an output of 11 then 01 must
be input. Also we can get an output of 11 with both 01 or 10 as inputs(angelic).

Figure 2.4: NDFA demonstrating angelic and demonic choices
Nondeterminism can also occur at the implementation level, typically in concurrent processing environments. The following discussion illustrates this. Nondeterministic program behavior can lead to the observation of different results in subsequent program runs based on the same input data. This kind of a problem is magnified in a parallel execution context due to several independent, but sporadically communicating tasks. In this case, a program may produce different results, although the same input data is provided and no modifications to the source code have been applied. Instead, the results of a program may depend on environmental factors, which cannot be controlled by the user (e.g. variations in processing speed, scheduling decisions, or contention in the network). While such nondeterministic behavior is already known in sequential programs, e.g. due to timing mechanisms, it is potentially significantly more severe in parallel environments with several independent tasks operating in loose cooperation.

The following scenario considers nondeterministic behavior in message passing programs, where receive operations are of major importance. The nondeterministic behavior may occur at wild card receive operations, non-blocking receives, and global operations, where the arrival order of messages may influence the program’s behavior in an undefined way. A call to a receive operation with a wild card as the source identifier allows a message from any process to be accepted. Consequently, the execution of a particular process after the receive operation may depend on the arrival order of messages at potential race condition. Consider three processes \(P, Q, R\) each communicating with the other. Assume that process \(P\) issues a receive operation with a wild card parameter, while both other processes issue a corresponding send. In this case,
which of the two execution scenarios occurs depends on which process’s send message
arrives first at $P$. To model such behavior in computer simulations, nondeterministic
behavior must be available as a feature for the software developer.
Chapter 3

Related Work

3.1 Binary Multirelations

Ingrid Rewitzky, proposed a new relational model, “Binary Multirelations”[9], to model program behavior in relational semantics. Binary Multirelations are relations which associate input elements to sets of states. The equivalence between relational models and predicate transformers[3] has been extended to show that any monotonic function over a Boolean algebra has an alternative representation as a multirelation[10]. The basic idea is that binary multirelations, being relations from states to sets of states, specify computations at the level of properties (with the sets of states) and at the level of states. So, multirelations can model demonic nondeterminism in terms of states at the level of the computations specified and angelic nondeterminism in terms of properties at the level of the specifications. The following concepts are from [5] and thus create parallel grounds from which the definitions and properties of our binary lifted multirelation are defined and derived.

In the standard relational model for programs, a binary relation specifies the input-output behavior in terms of the states that may be reached from a given state. Lifting
this description from the level of states to the level of properties, the behavior of a
program $\alpha$ may be specified in terms of the postconditions (or the properties) that it
has to specify.
Thus, if $\alpha$ has demonic choice, then
$sR_\alpha Q$ iff every execution of program $\alpha$, from the state $s$, reaches a state in which $Q$
is true.

or, if $\alpha$ has angelic choice, then
$sR_\alpha Q$ iff some execution of program $\alpha$, from the state $s$, reaches a state in which $Q$
is true.

$R_\alpha$ is, of course, different for the demonic case and the angelic case if $\alpha$ is truly
nondeterministic.

This representation of a specification computation $\alpha$ in terms of a binary multirelation
$R_\alpha$ relating states and postconditions is formalised as follows. From each starting
position $s$, the user may have the choice of more than one set of possible positions.
Therefore the set $\{Q | sR_\alpha Q\}$ captures the angelic choices available to the user. While
for each $Q$ with $sR_\alpha Q$, the set $\{t | t \in Q\}$ captures the demonic choices available
to the machine. Therefore a specification computation $\alpha$ may be represented as a
relation $R_\alpha$, the idea being that $\alpha$, when started in state $s$ is guaranteed to achieve
postcondition $Q$ for some angelic choice regardless of the demonic choice.

**Definition.** Let $X$ and $Y$ be sets. A binary multirelation on $X$ and $Y$ is a subset of
the Cartesian product $X \times \mathcal{P}(Y)$, that is a set of ordered pairs $(x, Q)$ where $x \in X$
and $Q \subseteq Y$. The image under a multirelation $R$ of any $x \in X$ is denoted by $R(x)$ and defined to be the set $\{Q \subseteq Y | xRQ\}$. In most cases $X = Y = S$.

**Composition.** Multirelational composition for any two binary multirelations $R \subseteq X \times \mathcal{P}(Y)$ and $T \subseteq X \times \mathcal{P}(Y)$ may be defined as follows,

$$R \circ T = \{(s, Q') | (\exists Q)[sRQ' \text{ and } Q' \subseteq \{y | yTQ\}]\}.$$  

So, given input value $s$, the angel can only guarantee that $R \circ T$ will achieve postcondition $Q$ if it can ensure that $R$ will establish some intermediate postcondition $Q'$ and if it can guarantee that $T$ will establish $Q$ given any value in $Q'$.

Note that relational composition for relations $r, t \subseteq S \times S$, namely

$$r; s = \{(x, z) | (\exists y)[xry \text{ and } ysz]\} = \{(x, z) | (\exists y)[xry \text{ and } y \in \{u | usz\}]\}$$
corresponds to multirelational composition.

**Relations** A relation is a fundamental mathematical concept expressing a relationship between elements of set. A binary relation is an arbitrary association of elements of one set with the elements of of another (or the same) set. In other words, a binary relation from a set $X$ to a set $Y$ is a subset $R \subseteq X \times Y$. So $R$ is a set of ordered pairs and $xRy$ means that $(x, y) \in R$. Sometimes, the concept of a “relation on the set $X$” is used which means that the relation is a subset of $X \times X$.

**Notation** Let $X$ and $Y$ be sets and let $R \subseteq X \times Y$ be a relation from $X$ to $Y$. For each $x \in X$, we let $xR = \{y \in Y | xRy\}$, and for each $y \in Y$, we let $Ry = \{x \in X | xRy\}$.

A few examples are,

1. Let $A = \mathbb{N}$, the set of natural numbers and for every $a, b \in A$, $aRb$ iff $a \leq b$.  

12
2. Let $A = \mathcal{P}(\mathbb{N})$ and for every $a, b \in A$, $aRb$ iff $a \cap b$ is finite.
3. Let $A = \mathcal{P}(\{1, \ldots, n\})$ and for every $a, b \in A$, $aRb$ iff $a \subseteq b$.

**Properties of relations.** For a relation on a set, there are many standard properties which may occur or hold.

A binary relation $R$ on set $A$ is,
1) reflexive, if for every $a \in X$, $aRa$. An example of such a relation is “is less than or equal to”, $x \leq y$ on $\mathbb{N}$.
2) symmetric, if for every $a, b \in X$, $aRb$ implies $bRa$. An example of such a relation is “is sibling of”.
3) antisymmetric, if for every $a, b \in X$, $aRb$ and $bRa$ implies $a = b$. An example of such a relation is “is greater than or equal to”, $x \geq y$.
4) asymmetric, if for every $a, b \in X$, $aRb$ implies $\neg(bRa)$. An example of such a relation is “is greater than”, $x > y$.
5) transitive, if for every $a, b, c \in X$, $aRb, bRc$ implies $aRc$. An example of such a relation is “is a factor of” over the set of integers.

**Operations.**

Inverse: If $R$ is a relation on $A \times B$, then $R^{-1}$ is a relation on $B \times A$ given by $R^{-1} = \{(b, a) | (a, b) \in R\}$.

Composition: The composition of relations $R_1 \subseteq A \times B$ and $R_2 \subseteq B \times C$ is the relation $R_1 \circ R_2 = \{(a, c) | (\exists b)((a, b) \in R_1 \text{ and } (b, c) \in R_2)\}$, i.e., $(a, c) \in R_1 \circ R_2$, if there exists an element $b$ such that the pair $(a, b)$ is in $R_1$ and the pair $(b, c)$ is in $R_2$. In other words, a “path” exists from element $a$ to element $c$ via some element in the set $B$. For example, composition of the “is parent-of” relation with itself gives the
relation “is grandparent-of”.

Union: The union of two relations $R, S \subseteq X \times Y$ is $R \cup S$, defined as $R \cup S = \{(x, y) | (x, y) \in R \text{ or } (x, y) \in S\}$.

Intersection: The intersection of two relations $R, S \subseteq X \times Y$ is $R \cap S$, defined as $R \cap S = \{(x, y) | (x, y) \in R \text{ and } (x, y) \in S\}$.

**Partial Order.** A binary relation $R \subseteq A \times A$ is a partial order if it is reflexive, transitive and asymmetric. A partial order is always defined on some set $A$ and is denoted by $\preceq$, $\leq$ or $\subseteq$. We often use the notation $\preceq$. The set along with the partial order defined on it is called a “poset” and is written as $(A, \preceq)$.
The following are a few examples of partial orders defined over the set $A = \mathbb{N}$.
1. The relation $\preceq=\leq$.
2. The relation $\preceq=\geq$.
3. The relation $\preceq=$ is factor of.
4. If $A = \mathcal{P}(\mathbb{N})$, $\subseteq\subseteq$ is a partial order over the set $A$.

### 3.2 Predicate Transformer Semantics and Nondeterminism

Predicate transformer semantics, first introduced in [4], is a method for defining the semantics of an imperative programming language. A predicate transformer is a total function from predicates on the state space (O) to predicates on the state space (I) of the program, where I = state space of input values and O = state space of output values. A *weakest precondition* wp, is a function, $wp : P(O) \rightarrow P(I)$. For a program $Q$, $\{A\}Q\{B\}$, is equivalent to a $wp(B) = A$, where B is a postcondition and A is a
precondition, i.e., \( B \in P(O) \) and \( A \in P(I) \). Now, say \( M \) is a deterministic program and \( R \) is its input-output relation, \( R \subseteq I \times O \). Since \( M \) is deterministic and \( R \) is a partial function, the weakest precondition of \( B \), \( wp(B) = \{ a \in I | aR \in B \} \) or \( \{ a \in I | \exists b \in B \text{ with } aRb \} \).

Let \( N \) be a nondeterministic program and \( R' \) be the associated relation, \( R' \subseteq I \times O \). There are now two natural predicate transformers we can define, \( t_a : P(O) \rightarrow P(I) \), for the angelic and demonic nondeterminism, respectively, where \( t_a(B) = \{ a \in I | aR \cap B \neq \emptyset \} \) and \( t_b(B) = \{ a \in I | aR \subseteq B \} \), for a postcondition \( B \). As an example, consider a program \( R \) defined over the state-space \( \{ X \in \mathbb{Z}^+ | 1 \leq X \leq 100 \} \) with a postcondition \( Q = \{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \} \) and

\[
\begin{align*}
R &= \{(1, 1), (1, 2), \ldots, (1, 8), (2, 2), (2, 3), \ldots, (2, 9), \\
&(3, 3), (3, 4), \ldots, (3, 10), (4, 4), \ldots, (4, 11), (5, 5), \ldots, (5, 12), \ldots\}.
\end{align*}
\]

Now, the weakest precondition, \( P_1 \), is the largest subset of input space, such that \( P_1(R)Q \). Considering the angelic and demonic nondeterminism by the program, \( t_a(Q) = \{ 1, 2, 3, \ldots 10 \} \) and \( t_d(Q) = \{ 1, 2, 3 \} \).

3.3 Galois Connections.

An order preserving Galois connection was first defined by in [11] by J.Schmidt is as follows.

**Definition** For partially ordered sets \((P, \sqsubseteq_P)\) and \((Q, \sqsubseteq_Q)\), a pair of functions \((f : P \rightarrow Q, g : Q \rightarrow P)\) is an order-preserving Galois connection between \( P \) and \( Q \) iff
(i) $f$ and $g$ are monotonic i.e., order-preserving and

(ii) for all $p \in P$, $p \sqsubseteq_P g f(p)$ and for all $q \in Q$, $f g(q) \sqsubseteq_Q q$. 
Chapter 4

Lifted Binary Multirelations

Binary multirelations are relations of the form $R \subseteq X \times \mathcal{P}(Y)$, between elements and sets of states. We now introduce the concept of lifted binary multirelations, which are an extension to the binary multirelations. Lifted binary multirelations are relations between two sets of states, i.e., of the form $R \subseteq \mathcal{P}(X) \times \mathcal{P}(Y)$.

4.1 Definition

Let $X$ and $Y$ be sets. A lifted binary multirelation on $X$ and $Y$ is a subset of the Cartesian product $\mathcal{P}(X) \times \mathcal{P}(Y)$, that is a set of ordered pairs $(A, B)$ where $A \subseteq X$ and $Q \subseteq Y$. In most cases $X = Y = S$.

Lifted binary multirelations enable us to use a relational approach to model program behavior. The image under a lifted binary multirelation $R$ of any $A \subseteq X$ is denoted $R(A)$ and defined to be the set $\{B \subseteq Y | ARB\}$.

Composition Multirelational composition for any two lifted binary multirelations $R \subseteq \mathcal{P}(X) \times \mathcal{P}(Y)$ and $T \subseteq \mathcal{P}(Y) \times \mathcal{P}(Z)$ is defined as follows,
\( R ; T = \{(A, C) | (\exists B') [ARB'] \text{ and } B' \in \{B | BRC\}\} \). So given a set of states, the angel can guarantee that \( R ; T \) will achieve a postcondition \( C \) if it can ensure that \( R \) will establish some intermediate postcondition \( B' \) and if it can guarantee that \( T \) will establish \( C \) given \( B' \).

The basis for our work here is that lifted binary multirelations are equivalent to binary multirelations, in the sense that binary multirelations "live" in the world of lifted binary multirelations. This is established using Galois connections in the next discussion. We use the following results about Galois connections.

**Proposition** Let \((f, (P, \subseteq_P), (Q, \subseteq_Q), g)\) be a Galois connection.

- \( f \) is surjective if and only if \( g \) is injective.
- \( f \) and \( g \) uniquely determine each other.

**Proof** See, for example, [6].

**Notation** Let \( X \) and \( Y \) be sets. Let \( A \) be the set of all relations from \( P(X) \) to \( P(Y) \) i.e., the set of all lifted binary multirelations; let \( B \) be the set of all binary multirelations from \( X \) to \( Y \) and let \( C \) be the set of all relations from \( X \) to \( Y \).

**Theorem** There exists Galois connections \((f_1, A, B, g_1), (f_2, B, C, g_2)\) and \((f_3, A, C, g_3)\), where the partial orders on \( A, B, C \) are subset inclusion. Further, each \( f_i \) is surjective; each \( g_i \) is injective; and \( f_2 \circ f_1 = f_3 \) and \( g_1 \circ g_2 = g_3 \).

**Proof:** Define \( f_1 : A \rightarrow B \) such that for \( R \subseteq P(X) \times P(Y) \), \((x, N) \in f_1(R) \) if and only if there exists \( M \subseteq X \) with \( x \in M \) and \((M, N) \in R \). Define \( g_1 : B \rightarrow A \) such that for \( S \subseteq X \times P(Y) \), \((M, N) \in g_1(S) \) if and only if \((m, N) \in S \) for each \( m \in M \). We define other maps similarly. For \( S \in B \), \((m, n) \in f_2(B) \) if and only if
there exists $N \subseteq Y$ such that $n \in N$ and $(m, N) \in S$. For $r \in C, (m, N) \in g_2(r)$ if and only if $(m, n) \in r$ for each $n \in N$. For $R \in \mathcal{A}, (m, n) \in f_3(R)$ if and only if there exists $M \subseteq X$ and $N \subseteq Y$ with $m \in M$, $n \in N$, and $(M, N) \in R$. For $r \in C$, $(M, N) \in g_3(r)$ if and only if for each $m \in M$ and for each $n \in N$, $(m, n) \in r$. Let $S_1, S_2 \in \mathcal{B}$ such that $S_1 \neq S_2$. Thus we may assume there exists $(m, N) \in S_1 - S_2$. It follows that $\{m\}, N \in g_1 S_1 - g_1 S_2$. Thus, $g_1$ is injective. Similarly, $g_2$ and $g_3$ are also injective. Thus, each $f_i$ is surjective. This result of $g_1$ being injective shows that binary multirelations “live” within the domain of lifted binary multirelations.

**Definition** A lifted binary multirelation $R \subseteq \mathcal{P}(X) \times \mathcal{P}(Y)$ is said to be up-closed in the second coordinate if $ARB$ and $B \subseteq B'$ together imply $ARB'$.

**Definition** A lifted binary multirelation $R \subseteq \mathcal{P}(X) \times \mathcal{P}(Y)$ is down-closed in the first coordinate if $ARB$ and $A' \subseteq A$ together imply $A'RB$.

The following is from [12].

**Definition** Let $R \in \mathcal{A}$. We say that $R$ has the union property if whenever $\mathcal{M} \subseteq \mathcal{P}(S)$ with $(M, N) \in R, \forall M \in \mathcal{M}$, then $(\bigcup \mathcal{M}, N) \in R$.

Let $\mathcal{B} \uparrow$ be the set of all up-closed binary multirelations on $S$ in $\mathcal{B}$. Also in [12], it has been shown that $R \in g_1(\mathcal{B} \uparrow)$ iff $R$ is down-closed in the first coordinate, up-closed in the second coordinate and has the union property. Since the results in [5], [9] and [10] assume up-closed binary multirelations when showing binary multirelations are equivalent to monotone predicate transformers, then the results obtained using binary multirelations can also be obtained using the images of the up-closed binary multirelations in the set of lifted binary multirelations.
**Notation** From now on, we represent a lifted binary multirelation as $R^*$ to distinguish from a binary multirelation which will be represented as $R$.

### 4.2 Angelic and Demonic Nondeterminism

In standard relational models for programs, a binary relation specifies the input-output behavior in terms of the states a program may reach from a given state. Lifting this description from the level of states to the level of properties, the behavior of a program $\alpha$ may be specified in terms of the postconditions (or the properties) that it has to specify, that is, if $\alpha$ has demonic choice then

$$AR^*_\alpha Q \text{ iff every execution of program } \alpha \text{ from a state in } A, \text{ reaches a state in which } Q \text{ is true.}$$

or, if $\alpha$ has angelic choice then

$$AR^*_\alpha Q \text{ iff some execution of program } \alpha \text{ from a state in } A \text{ reaches a state in which } Q \text{ is true.}$$

### 4.3 Refinement Ordering

There are two kinds of refinement orderings that can be defined over lifted binary multirelations. For any two lifted binary multirelations $R^*, T^* \subseteq \mathcal{P}(S) \times \mathcal{P}(S)$, the angelic refinement ordering $\sqsubseteq_\alpha$ is defined as follows

$$R^* \sqsubseteq_\alpha T^* \text{ iff } (\forall A \subseteq S), \{B \mid AR^* B\} \subseteq \{B \mid AT^* B\} \text{ iff } R^* \subseteq T^*.$$ The intuition here is that program $T^*$ has more angelic choice than $R^*$. 

20
Similarly, demonic refinement ordering can be defined for lifted binary multirelations at level of the computations and it is based on the set, $B$ related to a given state $a \in A \subseteq S$. So for $R^*, T^* \subseteq \mathcal{P}(S) \times \mathcal{P}(S)$,

$$R^* \sqsubseteq_d T^* \text{ iff } (\forall A \subseteq S)(\forall B \subseteq T^*(A))(\exists B' \subseteq R^*(A))[B' \subseteq B].$$

The refinement ordering must consist of a possible decrease in angelic choice and a possible increase in demonic choice.

There is an inconsistency in the way these orderings have been defined for binary multirelations in previous works. The following discussion details what might have been the motivation for these problems and our motivation in suggesting the solutions.

In [5] page 313, the angelic ordering is defined as

$$R \sqsubseteq_a T \text{ iff } R \supseteq T$$

and in [9], it is defined as

$$R \sqsubseteq_a T \text{ iff } R \subseteq T.$$

Our approach to solve this problem considers the way angelic nondeterminism is motivated. If $\alpha$ is a nondeterministic program, then the binary multirelation $R_\alpha$ represents angelic choice if $R_\alpha$ is defined as, $sR_\alpha Q$ iff some execution of $\alpha$ with input $s$ will produce a state in $Q$. Reasoning from the idea that $\alpha$ is more likely to succeed the larger the $R_\alpha$ is, then $R \sqsubseteq_a T$ iff $R \subseteq T$ seems the better definition. For the demonic ordering, we have for binary multirelations

$$R \sqsubseteq_d T \text{ iff } \forall s \in S \land \forall Q \in T(s), \exists Q' \in R(s) Q' \subseteq Q.$$

For up-closed multirelations, this condition is equivalent to $R \supseteq T$ and this implies that for a given $s \in S$, the demon is at least as likely to succeed under $T$ as under $R$ because the angel will need to choose a subset in $T(s)$ which is at least as large as
one which could have been chosen from $R(s)$.

We introduce two unique multirelations, the universal lifted binary multirelation $(\top^+)^*$ and the empty lifted binary multirelation $(\perp^+)^*$, defined as follows:

$$(\top^+)^* = \mathcal{P}(S) \times \mathcal{P}(S), \text{ i.e., } \{(A, B) | A \subseteq S \text{ and } B \subseteq S\}$$

$$(\perp^+)^* = \emptyset.$$  

The properties of these two lifted binary multirelations with respect to the angelic and demonic orderings are as follows,

$$(\top^+)^* \sqsubseteq_d R^* \text{ and } R^* \sqsubseteq_a (\top^+)^*, \text{ (}\top^+\text{)}^* \text{ is the worst lifted binary multirelation from the demon’s perspective and (}\top^+\text{)}^* \text{ is the best lifted binary multirelation from the angel’s perspective. The intuition here is that the angel may always choose the empty subset and the demon cannot stop this from happening. If we are considering a game as in the following discussion, then when the game is controlled by “(}\top^+\text{)” then the angel may always force the demon to lose on the move after the angel’s first move.}$$

Dually, for all the lifted binary multirelations $R^* \subseteq \mathcal{P}(S) \times \mathcal{P}(S)$,

$$(\perp^+)^* \sqsubseteq_a R^* \text{ and } R^* \sqsubseteq_d (\perp^+)^*, \text{ (}\perp^+\text{)}^* \text{ is the worst multirelation from the angel’s perspective since it offers the angel no choice for any } A \subseteq S \text{ and hence is the best multirelation from the demon’s perspective. Considering the same game specification as in the following scenario, when the game is “controlled” by (}\perp^+\text{)” the angel loses with its first move.}$$

We will now look at how to model two-player games using lifted binary multirelations. Assume $X = Y = S$, i.e., the lifted binary multirelations are built on single set $S$.  

22
Player A (for the angel) or player D (for the demon) may make the first move. A always chooses a predicate or subset of $S$ and D always chooses a subset of $S$. After A chooses a subset $Q$ of $S$, then D must choose an element in $Q$, and after D chooses an element $M \subseteq S$ in $Q$, then A must choose a subset which is related to $M$ via the lifted binary multirelation which is being used to direct the moves in the particular game in question. According to [10], a game is lost by a player if he/she is made to choose from the empty set. D can be given the empty set when A chooses to, as it is the primary goal of A to do so and win the game. However A is given the empty set if D chooses $M \subseteq S$ and if there are no subsets related to $M$ via the lifted binary multirelation “controlling ”the game. The previous discussion regarding $(\top^+)^*$ and $(\bot^+)^*$ seem to indicate that the first moves of the game should be randomly chosen. We cannot always force the angel or demon to make an initial move which will guarantee that the opponent may make a non-losing second move.

We now define the greatest lower bound(glb) and least upper bound(lub) with respect to the refinement orderings $\sqsubseteq_a$ and $\sqsubseteq_d$.

**Proposition** For any two lifted binary multirelations $R^*$ and $T^*$ defined over a set $\mathcal{P}(S)$ and where $R^*$ and $T^*$ are down-closed in the first coordinate, up-closed in the first coordinate and have the union property i.e., $R^*, T^* \in g_1(\mathcal{B} \uparrow)[12]$,

**Angelic Join** $R^* \sqcup_a T^* = R^* \cup T^*$

**Angelic Meet** $R^* \sqcap_a T^* = R^* \cap T^*$

**Demonic Join** $R^* \sqcup_d T^* = R^* \cap T^*$

**Demonic Meet** $R^* \sqcap_d T^* = R^* \cup T^*$

23
Claim $R^* \sqcup_a T^* = R^* \cup T^*$.

Proof: Let $Y^* \subseteq \mathcal{P}(S) \times \mathcal{P}(S)$ such that $Y^* = R^* \cup T^*$. Let $A \subseteq S$. Suppose that

$P_1 \in R^*(A)$ and $P_2 \in T^*(A)$. Then $P_1 \in Y^*(A)$, so $R^* \sqsubseteq_a Y^*$ and $P_2 \in Y^*(A)$, so

$T^* \sqsubseteq_a Y^*$. Thus $Y^*$ is an upper bound for $R^*$ and $T^*$.

Let $Z^* \subseteq \mathcal{P}(S) \times \mathcal{P}(S)$ and suppose that $Z^*$ is an upper bound for $R^*$ and $T^*$ with respect to $\sqsubseteq_a$. Let $A \subseteq S$, and let $P \in Y^*(A)$. By definition of $Y^*$, $P \in R^*(A)$ or $P \in T^*(A)$. Suppose without loss of generality that $P \in R^*(A)$. Since $R^* \sqsubseteq_a Z^*$, then $P \in Z^*(A)$. Hence $Y^* \sqsubseteq_a Z^*$ and $Y^*$ is the least upper bound for $R^*$ and $T^*$ with respect to $\sqsubseteq_a$, i.e., $Y^* = R^* \sqcup_a T^*$.

Claim $R^* \cap_a T^* = R^* \cap T^*$.

Proof: Let $Y^* \subseteq \mathcal{P}(S) \times \mathcal{P}(S)$ such that $Y^* = R^* \cap T^*$. Let $A \subseteq S$, suppose that

$Q \in Y^*(A)$. Since $Y^* \subseteq R^*$, then $Q \in R^*(A)$. By definition of angelic refinement, $Y^* \sqsubseteq_a R^*$. Similarly, $Y^* \sqsubseteq_a T^*$. Thus $Y^*$ is a lower bound.

Let $M^* \sqsubseteq_a R^*$ and $M^* \sqsubseteq_a T^*$. Suppose that $Q \in M^*(A)$. Since $M^* \sqsubseteq_a R^*$, then $Q \in R^*(A)$ and likewise $Q \in T^*(A)$. Therefore, $Q \in (R^* \cap T^*)(A) = Y^*(A)$. So $M^* \sqsubseteq_a Y^*$. Thus, $Y^*$ is the greatest lower bound.

Claim $R^* \sqcup_d T^* = R^* \cap T^*$

Proof: To show that $R^* \sqsubseteq_d R^* \cap T^*$

Let $Y^* \subseteq \mathcal{P}(S) \times \mathcal{P}(S)$ such that $Y^* = R^* \cap T^*$. Let $A \subseteq S$ and $Q \in Y^*(A)$. Since

$Y^* \subseteq R^*$, then $Q \in R^*(A)$ and $Q \subseteq Q$. By definition, $R^* \sqsubseteq_d Y^*$.

Let $P^* \subseteq \mathcal{P}(S) \times \mathcal{P}(S)$ such that $R^* \sqsubseteq_d P^*$ and $T^* \sqsubseteq_d P^*$. Let $A \subseteq S$ and $Q_p \in P^*(A)$. Since $R^* \sqsubseteq_d P^*$, then $\exists Q_r \in R^*(A)$ such that $Q_r \subseteq Q_p$. Since $R^*$ is up-closed then

$Q_p \subseteq R^*(A)$. Similarly, $Q_p \subseteq T^*(A)$. Thus, $Q_p \in (R^*(A) \cap T^*(A))$ and $Q_p \subseteq Q_p$. By
definition, $Y^*(A) \sqsubseteq_d P^*$, and $Y^*$ is the least upper bound.

Claim $R^* \sqcap_d T^* = R^* \cup T^*$

Proof: To show that $R^* \cup T^* \sqsubseteq_d R^*$.

Let $A \subseteq S$ and $Q \in R^*(A)$. Let $Y^* \subseteq \mathcal{P}(S) \times \mathcal{P}(S)$ such that $Y^* = R^* \cup T^*$. Since $R^*$, $Y^*$ are up-closed in the second coordinate, then $R^* \subseteq Y^*$, and $Q \subseteq Y^*(A)$. So $Y^* \sqsubseteq_d R^*$.

Let $P^*$ be a lifted binary multirelation such that $P^* \sqsubseteq_d R^*$ and $P^* \sqsubseteq_d T^*$. We need to show that $P^* \sqsubseteq_d Y^*$, i.e., $Y^*$ is the greatest lower bound.

Let $A \subseteq S$ and $Q \in Y^*(A)$. Without any loss of generality, assume $Q \in T^*(A)$. Since $P^* \sqsubseteq_d T^*$, $\exists Q' \in P^*(A)$ such that $Q' \subseteq Q$. By definition, $P^* \sqsubseteq_d Y^*$. Thus $R^* \sqcap_d T^* = R^* \cup T^*$, and it is the greatest lower bound with respect to $\sqsubseteq_d$.

4.4 Program Specifications Using Lifted Binary Multirelations

We now define a semantics of program and specification constructs in extensions [7,8,1] of Dijkstra’s guarded command language [3] using the operations defined in the previous sections. In [10], the same has defined in terms of binary multirelations.

We will be working with the set of lifted binary multirelations which are images of up-closed binary multirelations in the set of lifted binary multirelations i.e., $g_l(\mathcal{B} \uparrow)$. So $R^* \in g_l(\mathcal{B} \uparrow)$ iff $R^*$ is down-closed in the first coordinate, up-closed in the second co-ordinate and has the union property[12].

Definition

No operation $R^*_{\text{skip}} = \{(A, B) | A \subseteq B\}$.
Motivation: In [3], the semantics of the empty statement of a guarded command language, denoted by “skip”, are given by the definition that for any postcondition $R$, $wp(“skip” \cdot R) = R$. Also in [8], it is stated that the no-operation “skip” relates each state to itself. In [10], the skip binary multirelation, $R_{skip}$, is defined by $R_{skip} = \{(s, Q)|s \in S \text{ and } s \in Q\}$, where $Q \subseteq S$. In the binary multirelational model, $R_{skip}$ is the identity for composition. We first calculate $g_1(R_{skip})$, which we show to be $R_{skip}^\ast$. Then we show that $R_{skip}^\ast$ is the identity for composition in $g_1(B)$.

Claim $R_{skip}^\ast = g_1(R_{skip})$.

Proof: 1) To show that $g_1(R_{skip}) \subseteq R_{skip}^\ast$. Let $(A, B) \in g_1(R_{skip})$. By definition of $g_1$, $\forall a \in A, (a, B) \in R_{skip}$. So $\forall a \in A, a \in B$ i.e., $A \subseteq B$. By definition of $R_{skip}^\ast$, $(A, B) \in R_{skip}^\ast$. Therefore, $g_1(R_{skip}) \subseteq R_{skip}^\ast$.

2) To show that $R_{skip}^\ast \subseteq g_1(R_{skip})$. Let $(C, D) \in R_{skip}^\ast$. By definition, $C \subseteq D$. So $\forall c \in C, c \in D$ i.e., $\forall c \in C, (c, D) \in R_{skip}$. Therefore, by definition of $g_1(R_{skip})$, $(C, D) \in g_1(R_{skip})$. Therefore, $R_{skip}^\ast \subseteq g_1(R_{skip})$.

Hence $R_{skip}^\ast = g_1(R_{skip})$.

Claim $R_{skip}^\ast$ is the identity for $\vdash$ in $g_1(B)$.

Proof: 1) To show that $R_{skip}^\ast \vdash T^* \subseteq T^*$. Let $(A, B) \in R_{skip}^\ast \vdash T^*$. Then $\exists Q'$ such that $AR_{skip}^*Q'$ and $Q' \in \{B|BT^*Q\}$. So $A \subseteq B \in \{B|BT^*Q\}$. Therefore, $A \in \{B|BT^*Q\}$ because $T^*$ is down-closed in the first coordinate. It follows that $AT^*Q$. Hence $R_{skip}^\ast \vdash T^* \subseteq T^*$.

2) To show that $T^* \subseteq R_{skip}^\ast \vdash T^*$. Let $(A, C) \in T^*$. Since, $(A, A) \in R_{skip}^*$, then we have $AR_{skip}^*A$ and $A \in \{P \subseteq S|(P, C) \in T^*\}$. It follows that $(A, C) \in R_{skip}^\ast \vdash T^*$. Thus $T^* \subseteq R_{skip}^\ast \vdash T^*$ and hence $R_{skip}^\ast \vdash T^* = T^* \vdash R_{skip}^\ast = T^*$ is a two sided identity.
because $\;\uparrow$ is commutative.

Similarly, using $g_1$, one can show that

Divergence $R^*_{\text{abort}} = (\bot^+) = \emptyset$.  

Miracle $R^*_{\text{magic}} = (\top^+)$.  

Sequential Composition $R^*_{\alpha;\beta} = R^*_{\alpha} \ast R^*_{\beta}$.  

The following two results follow from the definitions of angelic and demonic joins.

Angelic Choice $R^*_{\alpha;\beta} = R^*_{\alpha} \sqcup_a R^*_{\beta}$.  

Demonic Choice $R^*_{\alpha;\beta} = R^*_{\alpha} \sqcup_d R^*_{\beta}$.  

Chapter 5

Conclusion and Future Work

The results shown in this thesis constitute a new calculus for modelling nondeterministic specifications using lifted multirelations. This methodology is similar to predicate transformers in the sense that they model programs at a “predicates” to “predicates” level or in other words “postconditions”. This lifted relational model is more expressive than the relational model and can model both angelic and demonic nondeterminism in a single framework. Refinement orderings and their associated properties can be used to define program and specification constructs of a guarded command language. Future work should consist of better understanding the equivalent to predicate transformers. Implementation models could possibly be in the areas of resource-sharing protocols and security protocols using refinement orderings in particular.
Bibliography


