The Metatheory of the Monadic Hybrid Calculus

by

Omar Alaqeeli
B.Sc., Qassim University, 2005
M.Sc., California State University, 2009

A Dissertation Submitted in Partial Fulfillment
of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

Supervisory Committee

The Metatheory of the Monadic Hybrid Calculus

by

Omar Alaqeeli
B.Sc., Qassim University, 2005
M.Sc., California State University, 2009

Supervisory Committee

Dr. William Wadge, (Department of Computer Science)
Supervisor

Dr. Bruce Kapron, (Department of Computer Science)
Departmental Member

Dr. Audrey Yap, (Department of Philosophy)
Outside Member

ABSTRACT

Supervisory Committee
Dr. William Wadge, (Department of Computer Science)
Supervisor
Dr. Bruce Kapron, (Department of Computer Science)
Departmental Member
Dr. Audrey Yap, (Department of Philosophy)
Outside Member

In this dissertation we prove the *Completeness*, *Soundness* and *Compactness* of the Monadic Hybrid Calculus $\mathcal{MHC}$ and we prove its expressive equivalence to the Monadic Predicate Calculus $\mathcal{MPC}$.

The Monadic Hybrid Calculus $\mathcal{MHC}$ is a new system that is based on the (propositional) modal logic *S5*. It is "Hybrid" in the sense that it includes quantifier free $\mathcal{MPC}$ and therefore, unlike S5, allows free individual constants. The main innovation in this system is the elimination of *bound variables.*

In $\mathcal{MHC}$, upper case letters denote properties and lower case letters denote individuals. Universal quantification is represented by square brackets, [ ], and existential quantification is represented by angled brackets, ⟨ ⟩. Thus, *All Athenians are Greek and mortal* is formalized as $[A](G \wedge M)$, *Some mortal Greeks are Athenians* as $\langle M \wedge G\rangle A$, and *Socrates is mortal and Athenian* as $s(M \wedge A)$.

We give the formal syntax and the formal semantics of $\mathcal{MHC}$ and give Beth-style Tableau Rules *(Inference Rules)*. In these rules, if $[\mathcal{P}]\mathcal{Q}$ is on the right then we select a new constant $\mathcal{v}$ and we add $\mathcal{v}\mathcal{P}$ on left, $\mathcal{v}\mathcal{Q}$ on the right, and we cancel the formula. If $[\mathcal{P}]\mathcal{Q}$ is on the left then we select a pre-used constant $\mathcal{p}$ and split the tree. We add $\mathcal{p}\mathcal{P}$ on the right of one branch and $\mathcal{p}\mathcal{Q}$ on the left of the other branch. We treat $\langle\mathcal{P}\rangle\mathcal{Q}$ similarly.

Our *Completeness* proof uses induction on formulas down a path in the proof tree. Our *Soundness* proof uses induction up a path. To prove that $\mathcal{MPC}$ is logically equivalent to the Monadic Predicate Calculus, we present algorithms that transform formulas back and forth between these two systems. *Compactness* follows immediately.

Finally, we examine the pragmatic usage of the Monadic Hybrid Calculus and we compare it with the Monadic Predicate Calculus using natural language examples. We also examine the novel notions of the Hybrid Predicate Calculus along with their pragmatic implications.

Table of Contents

List of Figures

ACKNOWLEDGMENTS

First and foremost, infinite thanks to my father and mother who are the main reason for me becoming the person I am now. They taught me well and with their unlimited love, kindness and support through all stages of my life I gained the confidence to pursue my studies and, indeed, with their encouragement and continuous praying I have been able to accomplished the higher levels of education. As promised, I made you proud and will continue further.

Limitless and countless thanks go to my wife, Hend. Thank you for your patience and the sacrifices you made for me. You have been by my side since the start and continue to be. Your patience when travelling with me on long flights back and forth home, your understanding of my long nights of studying, your absolute support for my ambitions in all of these years are all deeply appreciated.

Uncountable thanks go to my aunt for her kindness and continuous praying. I have definitely witnessed their effect on my life wherever and whenever I go and I know that this will continue for the rest of my life.

Enormous thanks go to my brothers and sisters who have been always respectful and willing to help in the time of need. Their wishes and prayers have influenced me in a way or another and assisted me to stay on the right path to reach this level of knowledge.

Immense thanks to my PhD supervisor, the distinguished logician, Dr. William Wadge. The moment I took the Logic and Artificial Intelligence course I was attracted by the fascination of Logic. I enjoyed solving the exercises and enhancing my knowledge in Logic as I wished to choose it to be my PhD dissertation topic, a wish I got. I have enjoyed working on this topic and benefited from your vast knowledge. Thank you for suggesting the topic and subtopics of my dissertation that I have immensely enjoyed working on. Thank you for the enormous meetings to advise, correct, comment and revise my work, as I wish that our mutual work will continue for years to come.

DEDICATION

To my beautiful, most beloved, daughters, Mayse and Shatha, your eyes give me hope, your smiles give me happiness, your laughter makes life worth living and your play makes working for the future a better choice. I hug your pure souls and love you indefinitely.

Chapter 1: Introductions

In this introductory chapter we give an overview of the Monadic Hybrid Calculus $\mathcal{MHC}$ system and the theorems that will be proven as well as other results produced in this dissertation. We also give a background for all logical systems that influenced our system along with some formulas' normalization methods and important definitions.

1.1. Introduction

In this dissertation we introduce the Monadic Hybrid Calculus $\mathcal{MHC}$ system and prove its *Completeness* and *Soundness* with respect to a Beth-style tableau proof system. We also prove that the $\mathcal{MHC}$ system is equivalent to the Monadic Predicate Calculus $\mathcal{MPC}$ system in expressive power. Thus, a *Compactness* proof will follow.

Informally, the syntax of $\mathcal{MHC}$ uses upper case letters to symbolize *properties* (one-argument predicates). For instance, in the property *"Apple trees that can grow in a cold climate",* the letter A can symbolize the property of being *an apple tree.* Similarly, the property *can grow in a cold climate* can be symbolized as the upper case letter C.

Lower case letters in $\mathcal{MHC}$ are used to symbolize *individuals* in natural or mathematical language statements. In a sentence like *"the apple in my hand is fresh"*, the individual *the apple in my hand* can be symbolized as lower case a but the property *fresh* is symbolized by an upper case letter F since it is a property not a particular individual. Thus, *"the apple in my hand is fresh"* is symbolized as aF. In another example, *"x is a plant", x* is a constant and P symbolizes the property *plant* so the formula will be xP. Similarly, *"x is an Apple that can grow in a cold climate"* can be symbolized as $x(A \wedge C)$ where the individual denoted by the constant *x* has both of the properties denoted by A and C.

The main syntactical difference between the $\mathcal{MHC}$ system and other systems is the quantifiers; square brackets, [ ], around a property formula for *universal* quantification and angled brackets, ⟨ ⟩, around a property formula for *existential* quantification. The two quantifiers are used to generalize the property enclosed over elements of the domain

of discourse that hold such property. For instance, if A and B are properties then $[A]B$ means that the property of being an A for *every* element in the domain of discourse makes such elements also have the property of being B. Also, if A and B are properties then $\langle A\rangle B$ means that *some* elements in the domain of discourse have the property of being an A and the property of being B. For instance, the sentence *"All apple trees can grow in a cold climate"* can be formalized as $[A]C$ thus for everything in the domain of discourse having the property of being an *Apple tree* makes it have the property *can grow in a cold climate*. Also, if the sentence is *"Some apple trees can grow in cold climate"* then the formula $\langle A\rangle C$ means that some elements in the domain of discourse have the property *Apple tree* and the property *can grow in cold climate*. By using [ ] and ⟨ ⟩ around properties we are not quantifying the properties themselves but we are restricting the quantification to only those individuals that have such properties.

The $\mathcal{MHC}$ quantifiers can formalize various forms of sentences depending on properties used in those sentences. For example, the sentence *"Some apple trees can grow in a cold climate and in a warm climate"* can be symbolized as $\langle A\rangle(C \wedge W)$ meaning *some* elements that have the property *an apple tree* have both the property *can grow in a cold climate* and the property *can grow in a warm climate.* Also, the $\mathcal{MHC}$ quantifiers, [ ] and ⟨ ⟩, can contains within them more than one property. For example, the sentence *"Some apple trees that can grow in a cold climate can also grow in a warm climate"* is symbolized in $\mathcal{MHC}$ system as $\langle A \wedge C\rangle W$ thus *some* element in the domain of discourse that are both *apple trees* and *can grow in cold climate* can *grow in warm climate.*

Expressions in $\mathcal{MHC}$ can be either *property formulas* or *absolute formulas.* The property formula is a Boolean combination of property constants. For example, if A and B are property formulas then their conjunction $A \wedge B$ is also a property formula. The absolute formulas, on the other hand, are either a combination of property formulas and constants; such as if A is a property formula and x is a constant then xA is an absolute formula, a combination of property formulas; such as if A and B are property formulas then $[A]B$, $\langle A\rangle B$ are absolute formulas, a Boolean combination of two or more absolute formulas, a Boolean negation of an absolute formula or truth constants, 0 and 1.

Like other logical systems, the $\mathcal{MHC}$ system uses the common notations for Boolean

connectives; ∧ for *conjunction*, ∨ for *disjunction*, ¬ for *negation*, → for *implication*. In $\mathcal{MHC}$, a Boolean connective can join either property formulas or absolute formulas. In a sentence like *"Apple trees can grow in either cold or warm climate"* we can form its corresponding formula as $[A](C \vee W)$ where A denotes *Apple tress* and C denotes the property *can grow in cold climate* that is joint with W which denotes the property *can grow in warm climate.* Also, the sentence *"Apple tress can not grow in sub-zero climate"* can be formalized as $[A]\neg S$ where A denotes *Apple tress* and $\neg S$ denotes the property *can not grow in sub-zero climate.*

The *negation* in the $\mathcal{MHC}$ system has the usual effect on the $\mathcal{MHC}$ quantifiers. A *universal* quantifier can be derived from an *existential* quantifier and contrariwise. Thus, $[A]B$ is logically equivalent to $\neg\langle A\rangle\neg B$ and $\langle A\rangle B$ is logically equivalent to $\neg[A]\neg B$.

The *properties*, *individuals* and *quantifiers* along with the *Boolean connectives* of the $\mathcal{MHC}$ system can formalize a wide range of logical statements. For instance, the sentence *"If Omar is a student then Bill is a professor"* can be formalized as $oS \rightarrow bP$ where o denotes the individual *Omar*, S is the property of being a *student*, b is the individual *Bill* and P is the property of being a *professor*. In another example, *"If all students are taking Arts and Biology then some students will be taking Chemistry or Math".* This sentence can be formularized as $[S](A \wedge B) \rightarrow \langle S\rangle(C \vee M)$ where S is the notion for the property of being a *student* and A, B, C and M are the properties of *Arts*, *Biology*, *Chemistry* and *Math* course, respectively.

In order to proceed with our proofs of the previously mentioned theorems, we have to introduce our Tableau Rules (Inference Rules) for the $\mathcal{MHC}$ system. These rules are based on the Beth-style tableaux method [1] where a vertical line separates formulas on the left that are presumed to be true and formulas on the right that are presumed to be false. Thus, our $\mathcal{MHC}$ Tableau Rules are as follows:

(In the following rules, $\mathcal{P}$ and $\mathcal{Q}$ are property formulas, $\mathcal{F}$ and $\mathcal{G}$ are absolute formulas and $\mathcal{v}$ and $\mathcal{p}$ are individual constants).

Note: in the following rules, $\mathcal{v}$s represent new constants and $\mathcal{p}$s represent previously used constants inside the proof tree where the tableau rules are applied.

I. $\langle \mathcal{P} \rangle \mathcal{Q}$ is *True*:

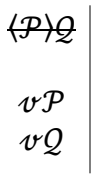

Figure 1.1: $\langle \mathcal{P} \rangle \mathcal{Q}$ is *True*

Let us assume we have the statement "*Some Apples are Red",* $\langle A \rangle R$, as a *true* statement. We first select a unique constant $\mathfrak{v}$ that has never been previously utilized inside the subject proof tree. We add each property with the selected constant as *true* statements and we eliminate the original formula, so that if $\langle A \rangle R$ is *true* and we chose *x* as our "fresh" constant then both $\mathfrak{x}A$, *x is an Apple,* and $\mathfrak{x}R$, *x is Red,* must be *true*.

II. $\langle \mathcal{P} \rangle \mathcal{Q}$ is *False*:

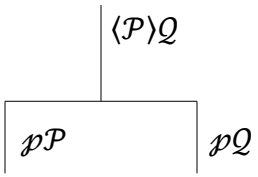

Figure 1.2: $\langle \mathcal{P} \rangle \mathcal{Q}$ is *False*

The tableau rule for $\langle A \rangle R$ as *false* however appears to be more complicated. In this tableau, we select a constant previously used inside the proof tree and we split into two branches. The first property, along with the selected constant, will be set up as *false* on the first branch and the other property as *false* on the second branch. Thus, either $\mathfrak{p}A$, *p is an Apple,* is *false* or $\mathfrak{p}R$, *p is Red,* is *false* will satisfy the condition of making $\langle A \rangle R$ *false*. We do not eliminate the original formula, $\langle A \rangle R$, thus we may have to use it repeatedly with other existing constants inside the proof tree. So, for every given constant in the proof we apply this rule until the proof is complete or all constants are exhausted (no more pre-used constants to select).

III. $[\mathcal{P}]\mathcal{Q}$ is *True*:

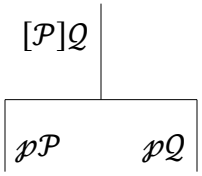

Figure 1.3: $[\mathcal{P}]\mathcal{Q}$ is *True*

The tableau rule for $[A]F$, *All Apples are Fruit,* as a *true* statement is similar to $\langle A\rangle R$ as a *false* statement in terms of its constant selection. We select a pre-used constant that appears in the proof tree and split into two branches. We set up the first property along with the selected constant as *false* on the first branch and the second property with its constant as *true* on the second branch. Thus either pA, *p is an Apple,* is *false* or pF, *p is a Fruit,* is *true*. We do not eliminate the original formula, $[A]F$, thus we may have to iterate its usage with other existing constants inside the proof tree. Hence, for every given constant in the proof we apply this rule until the proof is complete or all constants are exhausted.

IV. $[\mathcal{P}]\mathcal{Q}$ is *False*:

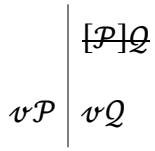

Figure 1.4: $[\mathcal{P}]\mathcal{Q}$ is *False*

The tableau rule for $[A]F$, *All apples are Fruits*, as *false* is simple. We choose a new constant v that is not used in the proof tree and set up the first property with this constant as *true* and the second property with this constant as *false*. Hence, xA, *x is an Apple,* is *true* and xF, *x is Fruit,* is *false* will simultaneously make $[A]F$, *All apples are Fruits*, *false*. We completely eliminate the original formula and never use it inside the proof tree afterward.

The previously listed Tableau Rules in conjunction with the usual Propositional Logic Tableau Rules are sufficient to test the validity of any $\mathcal{MHC}$ formula regardless of its complexity.

To illustrate these rules, let us take the following syllogism: $[A]B \wedge [B]C \rightarrow [A]C$. We will use the previously mentioned $\mathcal{MHC}$ tableau rules to verify it. Thus, the proof tree will be constructed as follows:

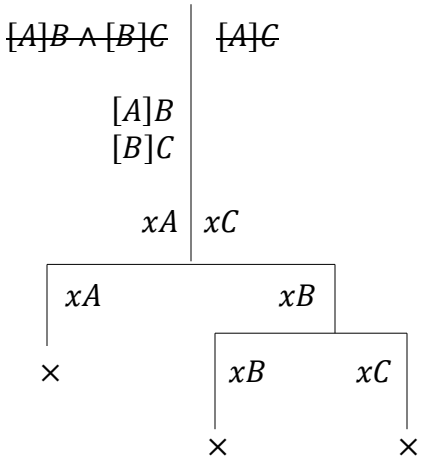

Figure 1.5: $\mathcal{MHC}$ Tableau Rules Example 1

When initiating the tableau proof tree (Beth-style) we locate the antecedent of the implication, $[A]B \wedge [B]C$, as *True* (on the left) and the consequence, $[A]C$, as *False* (on the right). We break up the conjunction $[A]B \wedge [B]C$ using the Propositional Logic tableau rule for conjunctions. Then, we apply the $\mathcal{MHC}$ Tableau Rule for $[A]C$ as *False* first so we can select a new constant, x, which can be used by other tableau rules. Thus, we will get xA as *True* and xC as *False*. Then, we apply the $\mathcal{MHC}$ tableau rule for $[A]B$ as *True* so we will have xA as *False* and xB as *True.* We also apply the same tableau rule for $[B]C$ as *True* so we will have xB as *False* and xC as *True.* Since all branches are closed the formula is valid.

In another example, let us take the following syllogism: $\langle A\rangle B \wedge [B]C \rightarrow \langle A\rangle C$ to be tested for logical validity. The proof tree is as follows:

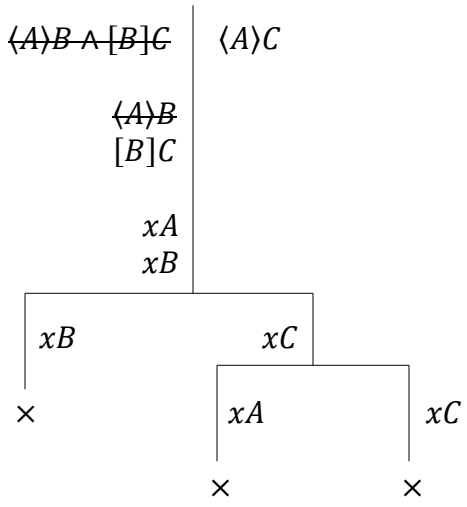

Figure 1.6: $\mathcal{MHC}$ Tableau Rules Example 2

We start by locating the antecedent of the implication, $\langle A\rangle B \wedge [B]C$, as *True* and the consequence, $\langle A\rangle C$, as *False*. We first break up the conjunction that appears in the antecedent using Propositional Logic tableau rule for conjunctions. We apply the $\mathcal{MHC}$ Tableau Rules for the $\langle A\rangle B$ as *True* first so we can choose a new constant, x, which can be reused when we resolve the other formulas. So, we will have xA and xB as *True*. We eliminate the formula $\langle A\rangle B$ so we will not use it afterward. We then apply the $\mathcal{MHC}$ Tableau Rule for $[B]C$ as *True* and form two branches where xB is *False* and xC is *True.* Finally, we apply the appropriate tableau rule for $\langle A\rangle C$ as *False* and form two new branches where both xA and xC are *False*. In every branch, there-exist a contradiction that closes the branch therefore the formula $\langle A\rangle B \wedge [B]C \rightarrow \langle A\rangle C$ is valid.

In the *Completeness* proof of the $\mathcal{MHC}$ system, we will assume that we have an assertion, $\delta_0, \delta_1, \delta_2, \ldots, \delta_{n-1} \rightarrow \psi$, and the tableau proof for this assertion has failed so that not all branches are closed and there-exists at least one branch open. We choose one open branch and we travel from the leaf to the root of this branch and as we travel we collect every formula of the form $\mathcal{v}\mathcal{V}$ ($\mathcal{v}$ is an individual constant and $\mathcal{V}$ is a property constant) found on the path to form an interpretation. Such an interpretation validates what exists on the left side of the open branch being *true* and what exists on the right side of the open branch being *false*. For absolute formulas appearing in the open branch as we

travel upward we use *induction* to prove that every formula on the left is *true* in the previously formed interpretation and every formula on the right is *false* in that interpretation.

In the *Soundness* proof, we assume that our tableau proof for our assertion, $\delta_0, \delta_1, \delta_2, \ldots, \delta_{n-1} \rightarrow \psi$, has succeeded so that all branches are closed. We argue that the assumption that there is a refutation leads to a contradiction. We show that for any node, if there is an interpretation that satisfies all the formulas on the left above the node and refutes all the formulas on the right above the node, then there is a node below with a similar refutation. It then follows that there is a path with refutations at each node on the path. But that means there is a refutation at the leaf of this path, impossible, since all paths are closed.

To strengthen our findings, we designed two algorithms, one that transforms formulas from the Monadic Predicate Calculus to the Monadic Hybrid Calculus and one that goes the other way. We also study the pragmatic usage of the $\mathcal{MHC}$ system and compare formulas in the Monadic Predicate Calculus with formulas in the Monadic Hybrid Calculus for the same group of logical statements.

The $\mathcal{MHC}$ *Compactness* proof uses model theory rather than proof theory. We assume that we have a set of $\mathcal{MHC}$ formulas such that all of its finite subsets are satisfiable. Every subset of this set has an equivalent form in the Monadic Predicate Calculus thus there is an equivalent set that is in the Monadic Predicate Calculus and it is finitely satisfiable. Consequently, any interpretation that is satisfiable in the Monadic Predicate Calculus set will be satisfiable in $\mathcal{MHC}$ set as well.

At this point, we expand our scope to examine the *General* Hybrid Predicate Calculus $\mathcal{HPC}$. The difference is that predicates can have more than one argument and these arguments can be quantifier phrases. For example, a statement with two arguments such as *Socrates likes Plato* can be formalized as spL where s denotes *Socrates*, p denotes *Plato* and L denotes the predicate *Like*. A statement with quantifier phrases like *Some Athenians like Socrates* can be formalized as $\langle A \rangle sL$ and a statement like *Every Athenian likes some philosopher* can be formalized as $[A]\langle P \rangle L$.

We explain the $\mathcal{HPC}$'s novel notions, $\sim$, $/$, $*$, and we present their logical implication and usage as well as give a detailed listing of the formal semantic that is

associated with each of these notions. The Hybrid Predicate Logic $\mathcal{HPC}$ system is vast and its *Completeness*, *Soundness* and *Compactness* have not been explored yet as well as its equivalency to First-Order Logic. In this dissertation, we explore the Monadic Hybrid Logic where predicates have only one argument.

The Hybrid Predicate Calculus is a system that exists in an area between Propositional Logic and Predicate Calculus but retains both of these systems' advantages and expressiveness. We believe that the Monadic Hybrid Calculus can formalize natural language more concisely and directly than Predicate Calculus.

1.2. Background

1.2.1. Formal Logic (0^{th} and 1^{st} Order Logic)

The purpose of Formal Logic is to formalize, using symbols, forms of Logic found originally in natural language. The power of expressiveness in each formal system varies depending on its degree (0^{th}, 1^{st}, 2^{nd} or higher). The Formal Logical system with the lowest degree is called Propositional Logic. It is often referred to as *Zero-Order* Logic since it does not allow for quantifiers to range over domains of discourse. For example, the sentence: "*If it rains, then flowers will bloom"* can be symbolized using Propositional Logic as: $R \rightarrow B$ where R denotes the antecedent of the sentence, *"If it rains", B* denotes the consequent of the sentence, *"flowers will bloom"*, and the arrow "→" denotes the Boolean connective; the *implication*.

The next degree of Formal Logic is the Predicate Logic, which is often referred to as *First-Order* Logic since it allows for quantifiers to range over individuals in the domain of discourse. For example, "*If it rains then some flowers will bloom".* The existential quantifier *some* indicates that it is not the case that whenever there is *rain* then any flower will *bloom* but rather it limits the consequent *bloom* to *some* flowers. Thus, the proposition in the sentence's antecedent is not absolute. The entire sentence can be symbolized as: $R \rightarrow (\exists x: F(x) \wedge B(x))$ where R denotes the proposition *it is raining,* $\exists$ is the existential quantifier for binding the individual x, F denotes the property of being a *flower* and B denotes the property *Bloom.* The symbol "∧" conjoins the two formulas

$F(x)$ and $B(x)$. The sentence can be rewritten as: *"If it rains then for some x, x is a flower and x blooms".*

The degree of Formal Logic continues further to *Second-Order* Logic. In *Second-Order* Logic, the quantifiers, unlike in *First-Order* Logic, range over both individuals and relations or properties of the domain of discourse which complicates the interpretation of its formulas. The Formal Logic degree can be extended even further to higher degrees, as there is no overhead limit. The higher the degree of the system the more complex its formulas become.

Several theorems emerged from the examination of different aspects of Formal Logic. One of these famous theorems is the *Completeness* theorem, which is one of the main theorems that are being studied in this dissertation. This theorem (for *First-Order* Logic) was developed by Kurt Gödel in 1939. It is the first attempt to create a relation between the syntax of formal systems and their semantics. In other words, creating a relationship between model theory and proof theory. In this dissertation, we are examining the theorem of *Completeness* along with other theorems, the *Soundness* and the *Compactness*, in the context of the Monadic Hybrid Calculus.

Every Formal Logic system consists of two parts: *Syntax* and *Semantics*. The *Syntax* is the study of the structure of formulas while the *Semantics* is the study of interpretations of these formulas. In the aforementioned example, "*If it rains then some flowers will bloom",* the *syntax* consists of the notions, $R \rightarrow (\exists x: F(x) \wedge B(x))$, and the *semantics* consists of a domain of discourse that has a set of *flowers* and a set of predicates that includes *Raining* and *Blooming.*

1.2.2. Modal Logic S5

Modal Logic is a branch of Formal Logic and an extension of Propositional Logic that deals with modality terms, such as *necessary* and *possible* [2] [3], that are not expressible by other logical systems. The *necessity* operator in Modal Logic is represented by a square $\Box$ while the *possibility* operator is represented by the diamond $\Diamond$. Since Modal Logic is an extension of other logical systems it uses similar notions for *variables*, *predicates* and *Boolean connectives*, in addition to the *necessity* and *possibility* operators.

For example,

"It is necessary that the sun will rise"

This sentence can be symbolized in Modal Logic as $\Box S$ where $\Box$ denotes the *necessity* term in the sentence and S is the predicate that is *the sun will rise.*

In another example,

"It is possible that rain will fail"

is represented in Modal Logic as $\Diamond R$ where $\Diamond$ denotes the *possibility* term and R is the predicate of the sentence *rain will fail.*

Modal Logic can symbolize more complex sentences. For example, sentences with Boolean *negation* can be symbolized as follows:

"It is not necessary that rain will fail": $\neg\Box R$

"It is not possible that the sun will not rise": $\neg \Diamond \neg S$

Also, some Modal Logics allow quantifiers as follows:

"For all x, if x is a flower and the sun is shining then it is possible that x will bloom"

$$\forall x(xF \wedge S \rightarrow \Diamond xB)$$

F is the property *flower, S* is the predicate *the sun is shining* and B is *will bloom.*

Modal Logic includes several systems that have different levels of complexity. The simplest among them is the Modal Logic S5 that includes the axioms:

- $\Box S \rightarrow S$
- $\Box\Box S \leftrightarrow \Box S$
- $\Box \Diamond S \leftrightarrow \Diamond S$

1.2.3. Hybrid Logic

Hybrid Logic [4] merges two logical systems into one. It is an extension of Modal Logic, which is itself an extension of Propositional Logic. In Hybrid Logic, new operands, *nominals* and @, were added to denote specific points in logical statements. For instance, in the sentence *"The sun will rise today at 6 o'clock"* we can formalize *"the sun will rise today"* by assigning a propositional variable, such as R, but the rest of the sentence, *"at 6 o'clock"*, is impossible to formalize. Here, the *nominals,* denoted by lower case letters, in conjunction with the *at* operand, @, are used to denote this part of the sentence. Thus, we

can formalize our sentence, *"the sun will rise today at 6 o'clock"*, as $@_xR$ where $@_x$ means *at point x*, *x* denotes "*6 o'clock"* and *R* denotes *"the sun will rise today"*. The operand $@_x$, @ along with the nominal *x*, are called the satisfaction operand. A formula like $@_x\mathcal{P}$ implies that *at* the very specific point *x* in the universe of discourse, the formula $\mathcal{P}$ is true.

Let us assume that we have the following sentence:

"If the sun goes down at 7 o'clock then the moon will rise at 8 o'clock"

This sentence can be formalized as $@_xS \rightarrow @_yM$ where $@_x$ denotes *"at 7 o'clock"*, S denotes *"the sun goes down"*, $@_y$ denotes *"at 8 o'clock"* and M denotes *"the moon will rise"*.

The Hybrid Logic also introduces two operands, ∀ and ↓, called *Binders*. These operands are used to bind *nominals* as follows: $\forall x\mathcal{P}$ means that $\mathcal{P}$ is true for all assignments to x, and $\downarrow x\mathcal{P}$ means that $\mathcal{P}$ is true of w if $\mathcal{P}$ is true when x is assigned to w.

Informally, the syntax of Hybrid Logic includes: $p, q, r, \ldots$ to denote the propositions of the sentence, $a, b, c, \ldots$ to denote the *nominals*, the @ operand to denote the *at*, □ and ◊, respectively, to denote *necessity* and *possibility*, *universal* quantifier ∀ and ↓ quantifier to bind the *nominals.*

Our system is inspired by both Modal Logic *S5* and Hybrid Logic. As we have seen, it bases the [ ] and ⟨ ⟩ notation on S5's □ and ◊ and the sG notation on $@_sG$. However, $\mathcal{MHC}$ is not literally an extension of these systems. In general, Modal Logic S5 and Hybrid Logic formulas are not syntactically $\mathcal{MHC}$ formulas.

$\mathcal{MHC}$ is hybrid in the sense that it combines aspects of S5 and quantifier free Monadic Predicate Calculus $\mathcal{MPC}$.

1.2.4. Formal Syntax

The Formal Syntax is the mathematical notation that a system uses to present logic statements as formulas. Such notation varies from one system to another. For example, the complete list of Propositional Logic notation includes: Boolean Connectives; ∧ for *conjunction*, ∨ for *disjunction*, → for *implication*, ¬ for *negation*, upper case letters

$\{A, B, C, \ldots\}$ for denoting *Properties*; and parentheses for combining two or more properties. In Predicate Calculus, the list extends to include: $\forall$ for *universal quantification,* $\exists$ for *existential quantification* and lower case letters $\{a, b, c, \ldots\}$ for denoting *variables* and *constants*.

To explain the Formal Syntax further, let us take a closer look at the syntax of the Hybrid Predicate Calculus. Informally, the syntax of the Hybrid Predicate Calculus $\mathcal{HPC}$ consists of upper case letters to denote *properties* and *predicates*, and a set of lower case letters (individual constants) to denote *individuals.* A sentence like *"Computer Logic is a required course"* can be formalized in $\mathcal{HPC}$ as cR where the lower case letter c denotes the individual *Computer Logic* and the upper case letter R denotes the predicate *a required course.*

Moreover, the meta-language of the $\mathcal{HPC}$ system uses an infinite set of circled numbers; …… ,⓶ ,①,⓪; to represent an ω-sequence of arguments that a *property* or *predicate* can possess. *Properties* and *predicates* in the $\mathcal{HPC}$ system are treated differently than in other logical systems. An important idea in the $\mathcal{HPC}$ system is to treat every *property* or *predicate* as having an infinite sequence of arguments. Thus, *properties* and *predicates* have infinite arity (more precisely, arity ω). For instance, if A is a *predicate* that takes one argument then it may be that A expresses *"⓪ is an Apple"*, if B is a *predicate* that takes two arguments then B may express *"① is a brother of ⓪"* and if C is a *predicate* that takes three arguments then C may express *"⓶ is taking the course ① with professor ⓪"*. We only assume the arity of the *property* or the *predicate* but we do not use the circled numbers when symbolizing the formula. For example, the sentence *"⓪ is an Apple"* is formalized as A. But if the sentence is *"x is an Apple"* then the formula is xA.

Similar to other logical systems, the $\mathcal{HPC}$ system uses Boolean connectives; $\wedge$ for *conjunction,* $\vee$ for *disjunction*, $\neg$ for *negation*, $\rightarrow$ for *implication*. For example, if we have the sentence *"If x is an Apple and y is an Orange then x and y are not from the same plant"* then we formalize it as $xA \wedge yO \rightarrow xy\neg S$ where A denotes the property *Apple,* $\wedge$ denotes the conjunction *and*, O denotes the property *Orange,* $\rightarrow$ denotes the implication *then*, $\neg$ denotes the negation *not* and S denotes the predicate *from the same plant.*

The syntax of the $\mathcal{HPC}$ system also includes the quantifiers; square brackets around

predicates, $[P]$, to denote a *universal quantifier*, angle brackets around predicates, $\langle P \rangle$, to denote an *existential quantifier*, that were introduced earlier.

In addition to the previously introduced syntactic elements, the $\mathcal{HPC}$ system introduces novel elements; $\sim$ for reversed relations, $/$ for reflexive relations, $*$ for numeral shifting (increment) of predicates' subjects and parentheses ( ) to assemble two or more predicates.

The new $\mathcal{HPC}$ notion *tilde*, $\sim$, is used to formalize passive voice in natural language. For instance, if T denotes *"① teaches ② "* the sentence can be "*Professor Wadge teaches Chemistry"* which is formed as wcT. When using $\sim$, the passive voice of the former sentence "*Chemistry is taught by professor Wadge"* will be formalized as $cw{\sim}T$ where c and w denote the individuals *Chemistry* and *professor Wadge,* respectively, $\sim T$ denotes the passive voice *taught by.* The new $\mathcal{HPC}$ notion *slash*, $/$, on the other hand, is used for reflexive verbs, such as in the sentence "*x likes him/herself"* that can be formalized as x/L which means xxL. Finally, the *star,* $*$*,* is used to substitute properties and predicates for *deduction* purposes. Thus, $bca*A$ is logically equivalent to bcA where * temporarily hides the constant a. In a later chapter, we will see how these notions are formally used when integrated with other $\mathcal{HPC}$'s notations.

1.2.5. Formal Semantics

The formal semantics is based on the notion of interpretation; a function that assigns meanings to symbols. For instance, let us assume that our domain of discourse is: *Members of the family*. In $\mathcal{MHC}$, for instance, a group of upper case letters will represent properties, such as A is the property of being a *Father* and B is the property of being a *Mother*. In First-Order Logic, another instance, a group of upper case letters will represent the relations between family members, such as A is the property of being a *Father*, and B is the *Mother-Child* relation, … etc. and a group of lower case letters will represent members of the family, such as x_0 is the father and x_1 is the older son … etc.
In our example, "*If it rains then some flowers will bloom",* the domain of discourse is *nature*. Thus, the *Raining, Flowering* and *Blooming* are the properties that are represented in upper case letters and $x_0, x_1, \ldots$ are the individuals in our domain of

discourse that possess the properties.

1.2.6. Predicate Calculus Normal Forms

A Predicate Calculus formula has various forms that it can be transformed to for specific purposes, as we will see in a further chapter. A normal form of a particular formula is its logically equivalent form but in a different syntactic order. The Prenex Normal Form (PNF) of a formula, φ, is it equivalent but in the form $\gamma_0, \gamma_1, \dots \gamma_n \mathcal{P}$ where each γ_i is either $\forall x$ or $\exists x$ and $\mathcal{P}$ is a quantifier-free formula.

The Conjunctive Normal Form (CNF) and the Disjunctive Normal Form (DNF) are other forms of normalization. A CNF of a formula, φ, is the *conjunction* of *disjunctions* of atomic formulas, or their negation, that gives φ the form $((\mathcal{Q}_0 \vee \mathcal{Q}_1 \vee \mathcal{Q}_2 \vee \dots \vee \mathcal{Q}_m) \wedge (\mathcal{Q}_{m+1} \vee \mathcal{Q}_{m+2} \vee \mathcal{Q}_{m+3} \dots) \wedge \dots)$. A DNF of a formula, φ, is the *disjunction* of *conjunctions* of atomic formulas, or their negation, that gives φ the form $((\mathcal{Q}_0 \wedge \mathcal{Q}_1 \wedge \mathcal{Q}_2 \wedge \dots \wedge \mathcal{Q}_m) \vee (\mathcal{Q}_{m+1} \wedge \mathcal{Q}_{m+2} \wedge \mathcal{Q}_{m+3} \dots) \vee \dots)$. In order to convert φ into its equivalent form of CNF or DNF, we may need to eliminate all Boolean *implications* that exist in the original formula, use the rules of distributions and propagate *negations* into clauses.

1.3. Definitions

Tautology*:* It literally means "repetition". The term used to describe a formula that is true in all interpretations. In other words, a statement that is true no matter how it is interpreted.

Consistency: Let S be a formula in the formal system A. We say that the system A is consistent if for every formula S there are not proofs for both S and $\neg S$.

Completeness: We say that the formal system $\mathcal{MHC}$ is complete if and only if for any set of axioms δ, if $\mathcal{I} \vDash \delta$ implies $\mathcal{I} \vDash \psi$ for any interpretation $\mathcal{I}$, then $\delta \vdash \psi$.

Soundness: Let δ be a set of axioms in the formal system $\mathcal{MHC}$. If ψ can be derived from δ, such that $\delta \vdash \psi$, then every interpretation $\mathcal{I}$ of δ, $\mathcal{I} \vDash \delta$, is also an interpretation of ψ, $\mathcal{I} \vDash \psi$.

Compactness: Let δ be an infinite set of axioms $\{\delta_0, \delta_1, \delta_2, \ldots\}$ in the formal system $\mathcal{MHC}$. If every finite subset of δ is consistent then δ is also consistent. Conversely, if δ is inconsistent then at least one of its finite subsets is inconsistent.

Chapter 2: The Monadic Hybrid Calculus $\mathcal{MHC}$

In this chapter, we start by explaining the pragmatic usage of the Monadic Hybrid Calculus $\mathcal{MHC}$ system and its ability to formalize sentences more effectively. Then, we give the formal syntax and formal semantics of $\mathcal{MHC}$. We also formally introduce the $\mathcal{MHC}$ tableau rules (inference rules) in Beth-style and explain them in full. For extra clarity, we also give the $\mathcal{MHC}$ tableau rules in the equivalent Smullyan-style that is widely used.

2.1. The Pragmatic Usage of the Monadic Hybrid Calculus $\mathcal{MHC}$

The pragmatic benefits of using the Monadic Hybrid Calculus $\mathcal{MHC}$ can be highlighted as follows: First, the $\mathcal{MHC}$ system does not allow *bound* variables. In formal logic where domains of discourse are involved to form interpretations, a hypothetical set of variables is presumed to be bound with properties, or relations. This set of variables has no actual equivalent elements listed in the statement being formalized. For example, *"All apples are either red or green".* In First-Order Logic, the syntax of this sentence will include a set of properties; *Apple, Red* and *Green*, and a set of variables to map these properties with them. Such a set, the set of variables, has no actual indication in the original sentence thus it increases the complexity of forming interpretations of the sentence's formula. The $\mathcal{MHC}$ system omits the usage of bound variables thus it significantly reduce the complexity of forming interpretations as well as formalizing sentences.

Another pragmatic benefit of using the Monadic Hybrid Calculus is that the $\mathcal{MHC}$ system is closer to natural language than other systems in two ways. First, properties in $\mathcal{MHC}$ are treated differently when symbolizing a sentence. For example, consider the following sentence: *"Tom is a father but not an engineer"*. This sentence is formalized in $\mathcal{MHC}$ as $t(F \wedge \neg E)$ but in *First-Order* Logic it is formalized as $tF \wedge \neg tE$. The first formula, the $\mathcal{MHC}$ formula, is closer to the sentence since the order of appearance of the individual t and the properties F and E is similar to their order of appearance in the original sentence while in the second formula it is not the case.

Second, quantifiers in the $\mathcal{MHC}$ system can enclose more than one property. For example, the sentence *"Some apples are green"* can be symbolized as $\langle A\rangle G$ where A denotes the property *apple* and G denotes the property *green*. But if the sentence is, for example, *"Some apples that are not grown locally are green"* then the symbolization will be $\langle A \wedge \neg L\rangle G$ where L denotes the property *local grown.* The word "*some"* in the sentence refers to any individuals in the domain of discourse that are both *apple* and *not grown locally* thus the quantification has to enclose those individuals who possess both of these properties. In *First-Order* Logic, the sentence *"Some apples that are not grown locally are green"* is symbolized as $\exists x(xA \wedge \neg xL \wedge xG)$ where the variable x appears once with every property in the antecedent of the formula which is slightly different than the sentence itself since the quantification binds the variable x that does not appear in the sentence. The *First-Order* Logic formula has the advantage of being symmetric in A, $\neg L$ and G. There is an $\mathcal{MHC}$ formula that is symmetric in the same way: $\langle A \wedge \neg L \wedge G\rangle 1$.

2.2. The Formal Syntax of the Monadic Hybrid Calculus $\mathcal{MHC}$

The Monadic Hybrid Calculus $\mathcal{MHC}$ uses the following symbols:

I. An infinite set of upper case letters $\{A, B, C, \ldots A_1, B_1, C_1, \ldots\}$ representing properties, *these are called property constants*;

II. An infinite set of lower case letters $\{x, y, z, \ldots x_1, y_1, z_1, \ldots\}$ representing individuals, *these are called individual constants*;

III. Logical symbols: $\wedge$ for *conjunction*, $\vee$ for *disjunction*, $\neg$ for *negation*, $\rightarrow$ for *implication*, square brackets around a property formula, $[\mathcal{P}]$, for *universal quantification*, angle brackets around a property formula, $\langle\mathcal{P}\rangle$, for *existential quantification* and parentheses ( ) to combine two or more formulas.

Expressions in $\mathcal{MHC}$ fall in two categories, *property formulas* and *absolute formulas.*

Definition 2.2.1. *A Monadic Hybrid Calculus property formula is a Boolean combination of property constants. In other words, either a property constant or an expression of the form:* $(\mathcal{P} \wedge \mathcal{Q}), (\mathcal{P} \vee \mathcal{Q}), (\mathcal{P} \rightarrow \mathcal{Q}), \neg\mathcal{P}, 0$ *or* 1, *where* $\mathcal{P}$ *and* $\mathcal{Q}$ *are property formulas.*

Definition 2.2.2. A Monadic Hybrid Calculus absolute formula is either:

i. $\mathcal{v}\mathcal{F}$, where $\mathcal{F}$ is a property formula and $\mathcal{v}$ is a constant. Such as: xA *"x is an apple",*

ii. $[\mathcal{A}]\mathcal{P}$, where $\mathcal{A}$ and $\mathcal{P}$ are both property formulas. Such as: $[S](P \wedge F)$ *"Anything that is a Strawberry is a Plant and a Fruit"*,

iii. $\langle\mathcal{A}\rangle\mathcal{P}$, where $\mathcal{A}$ and $\mathcal{P}$ are both property formulas. Such as: $\langle O\rangle(P \wedge F)$ *"Some things that are Organic are Plants and Fruit"*,

iv. A Boolean combination using logical symbols, namely $\wedge$ for *conjunction*, $\vee$ for *disjunction* or $\rightarrow$ for *implication*, to connect two or more formulas. Such that if $\mathcal{G}$ and $\mathcal{H}$ are formulas so are $(\mathcal{G} \wedge \mathcal{H}), (\mathcal{G} \vee \mathcal{H}), (\mathcal{G} \rightarrow \mathcal{H})$,

v. Boolean negation $\neg$, so that if $\mathcal{R}$ is a formula so is $\neg\mathcal{R}$. For instance, $\neg[A]R$, *"Not all Apples are Red",*

vi. Truth constants: 0 for *False* and 1 for *True.*

2.3. The Formal Semantics of the Monadic Hybrid Calculus $\mathcal{MHC}$

Definition 2.3.1. An interpretation $\mathcal{I}$ is a function that assigns to 1 a nonempty domain of discourse $\mathcal{I}(1)$-a nonempty set of individuals (people, things, days, …etc.), and that assigns to each property $\mathcal{V}$ a set $\mathcal{I}(\mathcal{V})$ of individuals and that assigns to each individual constant $\mathcal{v}$ an element $\mathcal{I}(\mathcal{v})$ of $\mathcal{I}(1)$.

$\mathcal{I} \vDash \mathcal{F}$ means, informally, that $\mathcal{F}$ is true given the interpretation $\mathcal{I}$. $\mathcal{I}, a \vDash \mathcal{P}$ means that $\mathcal{I}(a)$ has the property denoted by $\mathcal{P}$ given the interpretation $\mathcal{I}$.

Definition 2.3.2. The semantics of $\mathcal{MHC}$ formulas is given by the following rules (for all $\mathcal{MHC}$ absolute formulas, $\mathcal{F}$ and $\mathcal{G}$, and for all $\mathcal{MHC}$ property formulas $\mathcal{P}$ and $\mathcal{Q}$):

$\mathcal{I} \vDash \mathcal{v}\mathcal{P}$	*iff*	$\mathcal{I}, \mathcal{I}(\mathcal{v}) \vDash \mathcal{P}$
$\mathcal{I} \vDash [\mathcal{P}]\mathcal{Q}$	*iff*	for all a in $\mathcal{I}(1)$, if $\mathcal{I}, a \vDash \mathcal{P}$ then $\mathcal{I}, a \vDash \mathcal{Q}$
$\mathcal{I} \vDash \langle\mathcal{P}\rangle\mathcal{Q}$	*iff*	for some a in $\mathcal{I}(1)$, $\mathcal{I}, a \vDash \mathcal{P}$ and $\mathcal{I}, a \vDash \mathcal{Q}$
$\mathcal{I} \vDash \mathcal{F} \wedge \mathcal{G}$	*iff*	$\mathcal{I} \vDash \mathcal{F}$ and $\mathcal{I} \vDash \mathcal{G}$

$\mathcal{I} \vDash \mathcal{F} \vee \mathcal{G}$ *iff* $\mathcal{I} \vDash \mathcal{F}$ and/or $\mathcal{I} \vDash \mathcal{G}$

$\mathcal{I} \vDash \mathcal{F} \rightarrow \mathcal{G}$ *iff* It is not the case that: $\mathcal{I} \vDash \mathcal{F}$ and not $\mathcal{I} \vDash \mathcal{G}$

$\mathcal{I} \vDash \neg\mathcal{F}$ *iff* $\mathcal{I} \nvDash \mathcal{F}$

$\mathcal{I} \vDash 1$

$\mathcal{I} \nvDash 0$

Definition 2.3.3. The semantics of $\mathcal{MHC}$ property formulas is given by the following rules (for any $\mathcal{MHC}$ property formulas, $\mathcal{P}$ and $\mathcal{Q}$, any element a of $\mathcal{I}(1)$, and any property constant $\mathcal{V}$):

$\mathcal{I}, a \vDash (\mathcal{P} \wedge \mathcal{Q})$ *iff* $\mathcal{I}, a \vDash \mathcal{P}$ and $\mathcal{I}, a \vDash \mathcal{Q}$

$\mathcal{I}, a \vDash (\mathcal{P} \vee \mathcal{Q})$ *iff* $\mathcal{I}, a \vDash \mathcal{P}$ and/or $\mathcal{I}, a \vDash \mathcal{Q}$

$\mathcal{I}, a \vDash \mathcal{P} \rightarrow \mathcal{Q}$ *iff* It is not the case that: $\mathcal{I}, a \vDash \mathcal{P}$ and not $\mathcal{I}, a \vDash \mathcal{Q}$

$\mathcal{I}, a \vDash \neg\mathcal{P}$ *iff* $\mathcal{I}, a \nvDash \mathcal{P}$

$\mathcal{I}, a \vDash \mathcal{V}$ *iff* $a \in \mathcal{I}(\mathcal{V})$

$\mathcal{I}, a \vDash 1$

$\mathcal{I}, a \nvDash 0$

2.4. The Tableau Rules of the Monadic Hybrid Calculus $\mathcal{MHC}$

We are using Beth-style tableaux [1] with a vertical line separating the formulas presumed to be true on the left and the formulas presumed to be false on the right. Semantic Tableau Rules use refutation to prove the correctness of implications where premises appear on the left and are presumed to be true and their conclusion appears on the right and is presumed to be false. In Beth-style $\mathcal{MHC}$ Tableau Rules, we separate a quantified property from its conclusion property and we add free constants to each property. If there is a *Boolean connective*, we then use the well-known Propositional Logic Tableau Rules [1] to separate such a connective.

Definition 2.4.1. The Tableau Rules of $\mathcal{MHC}$ are as follows:
(In the following rules, $\mathcal{P}$ and $\mathcal{Q}$ are property formulas, $\mathcal{F}$ and $\mathcal{G}$ are absolute formulas, and v and p are individual constants).

I. $\langle \mathcal{P} \rangle \mathcal{Q}$ as *True*:

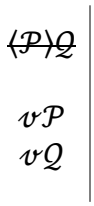

Figure 2.1: $\langle \mathcal{P} \rangle \mathcal{Q}$ as *True*

In this tableau rule, we first choose a new constant $\mathcal{v}$ that has never been previously used inside the proof tree where the tableau is applied and we add each property appearing in the original formula, along with the new constant, as *true*. Both have to be *true* to satisfy that $\langle \mathcal{P} \rangle \mathcal{Q}$ is *true.* We eliminate the original formula because we no longer need it afterward.

II. $\langle \mathcal{P} \rangle \mathcal{Q}$ as *False*:

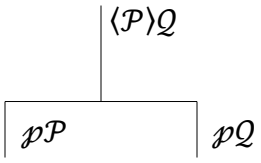

Figure 2.2: $\langle \mathcal{P} \rangle \mathcal{Q}$ as *False*

In the tableau rule of $\langle \mathcal{P} \rangle \mathcal{Q}$ as *False* we select a constant that has been previously used inside the proof tree where the tableau rule is applied and we split into two branches. The first property in the original formula, along with the selected constant, will be placed as *false* on the first branch (left) and the other property, along with the selected constant, will be placed as *false* on the second branch (right). This is because since $\langle \mathcal{P} \rangle \mathcal{Q}$ is *false*, $\mathcal{p}$ cannot be a witness to this formula. Thus $\mathcal{p}$ either does not have the property $\mathcal{P}$ or does not have the property $\mathcal{Q}$. In this rule we do not eliminate the original formula because we may have to reuse it with other existing constants inside the proof tree therefore for every constant appearing in the tree proof we apply this rule until all constants are exhausted.

Note: In rare cases where no constants are being used inside the proof tree, such as where we have only this case to apply, we simply select a new constant.

III. $[\mathcal{P}]\mathcal{Q}$ as *True*:

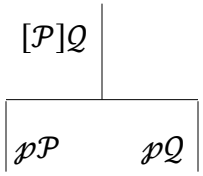

Figure 2.3: $[\mathcal{P}]\mathcal{Q}$ as *True*

$[\mathcal{P}]\mathcal{Q}$ as *True* is quite similar to the previous rule. We select a constant that has been previously used inside the proof tree where the tableau rule is applied and we split it into two branches. The first property in the original formula, along with the selected constant, will be placed as *false* on the first branch (left) and the other property, along with the selected constant, will be placed as *true* on the second branch (right). Since p cannot be a witness to the falsity of $[\mathcal{P}]\mathcal{Q}$, either p has the property $\mathcal{Q}$ or fails to have the property $\mathcal{P}$. Also, in this rule we do not eliminate the original formula so we can reuse it with other existing constants inside the proof tree, if any, until all constants are exhausted.

Note: As with the $\langle\mathcal{P}\rangle\mathcal{Q}$ rule, in rare cases where no constants are being used inside the proof tree, we have no option but to select a new constant.

IV. $[\mathcal{P}]\mathcal{Q}$ as *False*:

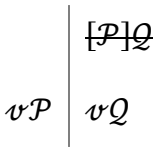

Figure 2.4: $[\mathcal{P}]\mathcal{Q}$ as *False*

This rule is quite similar to the first tableau rule introduced. If $[\mathcal{P}]\mathcal{Q}$ is *false* we first choose a new constant v that has never been previously used inside the proof tree where the rule is applied and we add the generalized property in the original formula, along with the new constant, as *true* and the other property, along with the new constant, as *false.*

Thus, the new formulas together assert that v is a witness to the falsity of $[\mathcal{P}]\mathcal{Q}$. Finally, we eliminate the original formula because we no longer need to use it afterward.

Rules of distribution for common factor constants over properties such as in $x(P \wedge Q) \equiv xP \wedge xQ$ are applicable in case we have joint properties denoted for an individual.

The previously listed tableau rules are not sufficient for the simplest Boolean combination of atomic formulas. Thus we need to include in our proof the well-known propositional logic tableau rules to eliminate Boolean connectives.

The following tableau rules are available in any $\mathcal{MHC}$ proof:

i. Negation:

Figure 2.5: Rule of Negation

Let A be a formula, if $\neg A$ is *true*, then A is *false*. Likewise, if $\neg A$ is *false* then A is *true*.

ii. Conjunction:

Figure 2.6: Rule of Conjunction

If a Boolean conjunction, such as $A \wedge B$ is *true* then A and B are both *true*. If such conjunction appears as *false* then either A or B is *true* so we split into two branches, one branch for each possibility.

iii. Disjunction

Figure 2.7: Rule of Disjunction

If the disjunction of A and B is *true*, then either A or B is *true* will satisfy making the disjunction *true* thus we have to split into two branches. If the disjunction of A and B is *false* then both A and B must be *false*.

iv. Implication

Figure 2.8: Rule of Implication

If a logical implication, such as in $A \rightarrow B$, appears as *true* then either A is *false* or B is *true* will prove that $A \rightarrow B$ is *true* so that we have to split into two branches. If $A \rightarrow B$ is *false* then A must be *true* and B must be *false*.

Any formula that is a combination of $\mathcal{MHC}$ formulas regardless of its complexity can be disassembled using these tableau rules.

Definition 2.4.2. *A successful tableau proof is a tableau with all its branches closed.*

A branch is closed if the same formula appears on *both* sides of the branch and a branch is open if there are no further tableau rules to apply and yet there is no formula that appears on *both* sides of the branch. If we have a set δ of axioms and a conclusion ψ we use a tableau that at its root has δ on the left and ψ on the right. If the tableau succeeds (all its branches closed) then we can assert that $\delta \cup \{\neg\psi\}$ is inconsistent and thus $\delta \vdash \psi$.

A significant number of Logic books use a tableau method that is based on Smullyan-style where a formula's components are resolved in a hierarchal order. We present the following $\mathcal{MHC}$ tableau rules using Smullyan-style that are logically equivalent to the aforementioned $\mathcal{MHC}$ tableau rules that use Beth-style.

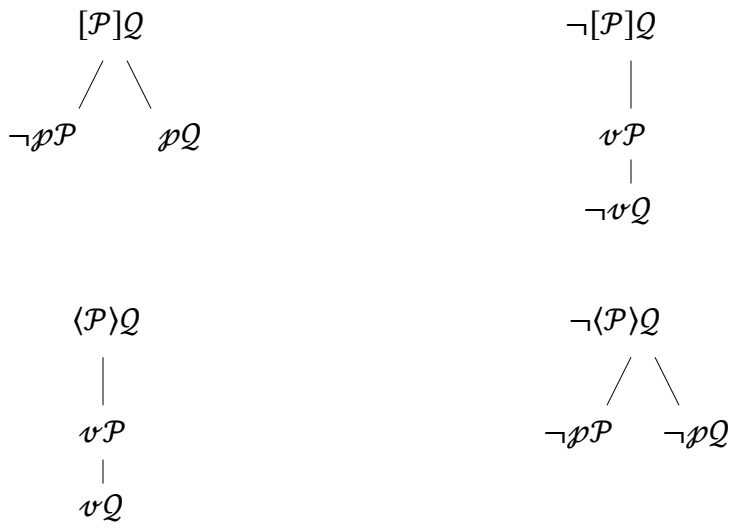

Figure 2.9: $\mathcal{MHC}$ Tableau Rules in Smullyan-style

Chapter 3: The Completeness of the Monadic Hybrid Calculus $\mathcal{MHC}$

In this chapter, we present the proof of Completeness of the Monadic Hybrid Calculus $\mathcal{MHC}$ with respect to our tableau rules. We first give our definitions of the Completeness along with the necessary theorems and then give a tableau proof example.

3.1. A Completeness Proof for the Monadic Hybrid Calculus $\mathcal{MHC}$

We shall use $\vDash$ as the satisfaction symbol, so that $\mathcal{I} \vDash \delta$ means, for instance, that the interpretation $\mathcal{I}$ satisfies the set of formulas δ by satisfying every element of δ. We use $\vdash$ as the derivation symbol, so that $\delta \vdash \psi$ means, for instance, the conclusion ψ can be derived from the set of axioms δ using a tableau.

Definition 3.1.1 *(Completeness)*: We say that the formal system $\mathcal{MHC}$ is complete if and only if for any finite set of formulas δ and any formula ψ, if $\mathcal{I} \vDash \delta$ implies $\mathcal{I} \vDash \psi$ for any interpretation $\mathcal{I}$, then $\delta \vdash \psi$.

If ψ can be derived from the finite set of axioms δ then the tableau proof of the assertion $\delta_0, \delta_1, \delta_2, \ldots \delta_{n-1} \rightarrow \psi$ will definitely halt at some point and all branches will be closed. But if ψ can not be derived from the finite set of axioms δ then the tableau proof of the assertion $\delta_0, \delta_1, \delta_2, \ldots \delta_{n-1} \rightarrow \psi$ will never halt thus at least one branch will still be open.

The completeness proof of the Monadic Hybrid Calculus $\mathcal{MHC}$ with respect to the previously introduced tableau rules is as follows:

Theorem 3.1.2. *The formal system $\mathcal{MHC}$ is complete.*

Proof 3.1.3. Let $\delta_0, \delta_1, \delta_2, \ldots \delta_{n-1} \rightarrow \psi$ be an assertion, where δ is set of axioms, whose tableau proof fails. We will show that there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \vDash \delta$ and $\mathcal{I} \nvDash \psi$.

We can assume that the corresponding tableau for $\delta_0, \delta_1, \delta_2, \ldots, \delta_{n-1} \rightarrow \psi$ has at least one branch open. Thus, for each open branch, there is a counterexample residing in such a branch. To find out this counterexample, we trace the path within the open branch from leaf to root and we collect every formula of the form $\mathcal{v}\mathcal{V}$ residing in such path.

Thus, for every atomic formula of the form $\mathcal{v}\mathcal{V}$, where $\mathcal{v}$ is any individual constant and $\mathcal{V}$ is any property constant, in the open branch we presume what appears on the *left* side of the open branch is *True* and what appears on the *right* side is *False.* Such that,

$$\mathcal{v}\mathcal{V} = T \quad \textit{iff} \quad \mathcal{v}\mathcal{V} \text{ appears on the } \textit{left}$$
$$\mathcal{v}\mathcal{V} = F \quad \textit{iff} \quad \mathcal{v}\mathcal{V} \text{ appears on the } \textit{right}$$

We collect all formulas of this type that appear on the open branch and we construct a counterexample that represents an interpretation $\mathcal{I}$, such that $\mathcal{I} \vDash \delta$ but $\mathcal{I} \nvDash \psi$.

Let T be the tableau under examination that has O as an open branch. For constructing $\mathcal{I}$, let $\{x, y, z, \ldots\}$ be the free constants on either side of the open branch and take this set to be the universe of discourse of $\mathcal{I}$. Let $\mathcal{I}(A)$, $\mathcal{I}(B)$, $\mathcal{I}(C)$, … be the semantic interpretations in this universe where $A, B, C, \ldots$ are any possible properties residing in the open branch and associated with the universe of discourse.

To construct $\mathcal{I}$, choose $\mathcal{I}(A)$, $\mathcal{I}(B)$, $\mathcal{I}(C)$, … subject to the following constraints:

$$\mathcal{v} \in \mathcal{I}(\mathcal{V}) \qquad \textit{iff} \quad \mathcal{v}\mathcal{V} \text{ appears on the } \textit{left} \text{ side of the open branch}$$
$$\mathcal{v} \notin \mathcal{I}(\mathcal{V}) \qquad \textit{iff} \quad \mathcal{v}\mathcal{V} \text{ appears on the } \textit{right} \text{ side of the open branch}$$

This is possible because we will never find $\mathcal{v}\mathcal{V}$ on both sides of the open branch, otherwise the branch will be closed.

The atomic formulas collected above are for formulas that have no quantifiers or logical connectives. For those with connectives, we prove by induction as we travel up

the open branch that any formula on the left is *true* and any formula on the right is *false* in $\mathcal{I}$.

Let $\mathcal{F}$ and $\mathcal{G}$ be formulas on the open branch O. For every given formula along O, one of the following cases may occur:

I. $\langle\mathcal{F}\rangle\mathcal{G}$ *on the left:*

Let $\langle\mathcal{F}\rangle\mathcal{G}$ be on the left side of the open branch, then somewhere below in the proof tree we have $v\mathcal{F}$ and $v\mathcal{G}$ on the left side of the open branch. Our open branch will continue on the side where by induction both $v\mathcal{F}$ and $v\mathcal{G}$ are *true* therefore $\langle\mathcal{F}\rangle\mathcal{G}$ is *true*.

In other words,

$v\mathcal{F} \Rightarrow 1$ and $v\mathcal{G} \Rightarrow 1$
$\therefore \langle\mathcal{F}\rangle\mathcal{G} \Rightarrow 1$

O

$\langle\mathcal{F}\rangle\mathcal{G}$

$v\mathcal{F}$
$v\mathcal{G}$

Figure 3.1: $\langle\mathcal{F}\rangle\mathcal{G}$ on the *left* of the branch

II. $\langle\mathcal{F}\rangle\mathcal{G}$ *on the right:*

Let $\langle\mathcal{F}\rangle\mathcal{G}$ be on the right of the open branch, then somewhere below in the proof tree we have a branch split with $x\mathcal{F}$ or $y\mathcal{F}$ or $z\mathcal{F}$…etc. (since the tableau rule for resolving $\langle\mathcal{F}\rangle\mathcal{G}$ required exhausting all possible constants that exist in the proof) on the right of the left branch and $x\mathcal{G}$ or $y\mathcal{G}$ or $z\mathcal{G}$ …etc., respectively, on the right of the right branch. Our open branch must continue on either one of these splits so by induction either $x\mathcal{F}$ is *false* or $x\mathcal{G}$ is *false* and either $y\mathcal{F}$ is *false* or $y\mathcal{G}$ is *false*, … etc., therefore $\langle\mathcal{F}\rangle\mathcal{G}$ is *false*.

In other words, for any given constant x *or* y *or* z ... :

$x\mathcal{F} \Rightarrow 0$ or $x\mathcal{G} \Rightarrow 0$

$y\mathcal{F} \Rightarrow 0$ or $y\mathcal{G} \Rightarrow 0$

…

…

$\therefore \langle\mathcal{F}\rangle\mathcal{G} \Rightarrow 0$

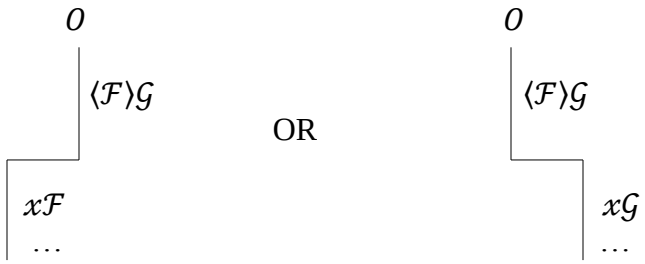

Figure 3.2: $\langle\mathcal{F}\rangle\mathcal{G}$ on the *right* of the branch

Note that we assume that during forward proof (top-to-base proof) we completely exhausted all possible constants that are identical to constants that already exist in order to refute any atomic formula that holds the same property.

III. $[\mathcal{F}]\mathcal{G}$ *on the left:*

Let $[\mathcal{F}]\mathcal{G}$ be on the left of the open branch, then somewhere below in the proof tree we have a branch split with $x\mathcal{F}$ or $y\mathcal{F}$ or $z\mathcal{F}$…etc. (since the tableau rule for $[\mathcal{F}]\mathcal{G}$ required exhausting all possible constants that are already exist inside the proof tree) on the right of the left branch and $x\mathcal{G}$ or $y\mathcal{G}$ or $z\mathcal{G}$ or…etc., respectively, on the left of the right branch. In each case, our open branch must continue on one of the branch splits so by induction either $x\mathcal{F}$ is *false* or $x\mathcal{G}$ is *true*, and either $y\mathcal{F}$ is *false* or $y\mathcal{G}$ is *true*, … etc., therefore $[\mathcal{F}]\mathcal{G}$ is *true*.

In other words, for any given constant x *or* y *or* z ... :

$x\mathcal{F} \Rightarrow 0$ or $x\mathcal{G} \Rightarrow 1$

$y\mathcal{F} \Rightarrow 0$ or $y\mathcal{G} \Rightarrow 1$

…

…

$\therefore [\mathcal{F}]\mathcal{G} \Rightarrow 1$

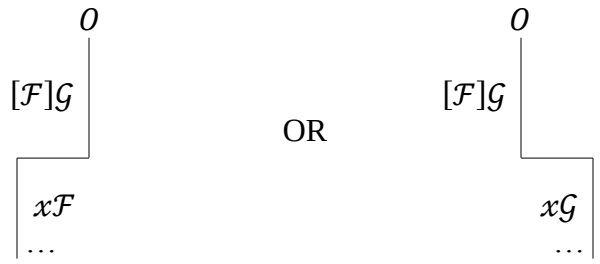

Figure 3.3: $[\mathcal{F}]\mathcal{G}$ on the *left* of the branch

We also here assume that during the forward proof (top-to-base proof), all possible constants existing in the proof tree are already exhausted thus all atomic formulas have been refuted.

IV. $[\mathcal{F}]\mathcal{G}$ *on the right:*

Let $[\mathcal{F}]\mathcal{G}$ be on the right side of the open branch, then somewhere below in the proof tree we have $\mathcal{v}\mathcal{F}$ on the left and $\mathcal{v}\mathcal{G}$ on the right side of the open branch. Our open branch will continue on where by induction $\mathcal{v}\mathcal{F}$ is *true* and $\mathcal{v}\mathcal{G}$ is *false* therefore $[\mathcal{F}]\mathcal{G}$ is *false*.

In other words,

$\mathcal{v}\mathcal{F} \Rightarrow 1$ and $\mathcal{v}\mathcal{G} \Rightarrow 0$
$\therefore [\mathcal{F}]\mathcal{G} \Rightarrow 0$

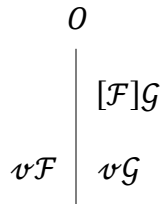

Figure 3.4: $[\mathcal{F}]\mathcal{G}$ on the *right* of the branch

V. Conjunction on the left:

Let $\mathcal{F} \wedge \mathcal{G}$ be on the left side of the open branch, then somewhere below in the proof tree we have $\mathcal{F}$ and $\mathcal{G}$ on the left side of the open branch. Our open branch will continue where by induction both $\mathcal{F}$ and $\mathcal{G}$ are *true* therefore $\mathcal{F} \wedge \mathcal{G}$ is *true*.

In other words,

$$\mathcal{F} \Rightarrow 1$$
$$\mathcal{G} \Rightarrow 1$$

$$\therefore \mathcal{F} \wedge \mathcal{G} \Rightarrow 1$$

VI. *Conjunction on the right:*

Let $\mathcal{F} \wedge \mathcal{G}$ be on the right side of the open branch, then somewhere below in the proof tree we have a branch split with $\mathcal{F}$ on the right of the left branch and $\mathcal{G}$ on the right of the right branch. Our open branch must continue on one of these splits so by induction either $\mathcal{F}$ or $\mathcal{G}$ is *false* therefore $\mathcal{F} \wedge \mathcal{G}$ is *false*.

In other words,

$$\mathcal{F} \Rightarrow 0 \text{ or } \mathcal{G} \Rightarrow 0$$

$$\therefore \mathcal{F} \wedge \mathcal{G} \Rightarrow 0$$

VII. *Disjunction on the left:*

Let $\mathcal{F} \vee \mathcal{G}$ be on the left side of the open branch, then somewhere below in the proof tree we have a branch split with $\mathcal{F}$ on the left of the left branch and $\mathcal{G}$ on the left of the right branch. Our open branch must continue on one of these splits so by induction either $\mathcal{F}$ or $\mathcal{G}$ is *true* therefore $\mathcal{F} \vee \mathcal{G}$ is *true*.

In other words,

$$\mathcal{F} \Rightarrow 1 \text{ or } \mathcal{G} \Rightarrow 1$$

$$\therefore \mathcal{F} \vee \mathcal{G} \Rightarrow 1$$

VIII. *Disjunction on the right:*

Let $\mathcal{F} \vee \mathcal{G}$ be on the right side of the open branch, then somewhere below in the proof tree we have $\mathcal{F}$ and $\mathcal{G}$ on the right side of the open branch. Our open branch will continue where by induction both $\mathcal{F}$ and $\mathcal{G}$ are *false* therefore $\mathcal{F} \vee \mathcal{G}$ is *false*.

In other words,

$$\mathcal{F} \Rightarrow 0$$

$$\mathcal{G} \Rightarrow 0$$

$$\therefore \mathcal{F} \vee \mathcal{G} \Rightarrow 0$$

IX. *Implication on the left:*

Let $\mathcal{F} \to \mathcal{G}$ be on the left side of the open branch, then somewhere below in the proof tree we have a branch split with $\mathcal{F}$ on the right of the left branch and $\mathcal{G}$ on the left of the right branch. Our open branch must continue on either one of these splits so by induction either $\mathcal{F}$ is *false* or $\mathcal{G}$ is *true* therefore $\mathcal{F} \to \mathcal{G}$ is *true*.

In other words,

$$\mathcal{F} \Rightarrow 0 \text{ or } \mathcal{G} \Rightarrow 1$$
$$\therefore\ \mathcal{F} \to \mathcal{G} \Rightarrow 1$$

X. *Implication on the right:*

Let $\mathcal{F} \to \mathcal{G}$ be on the right side of the open branch, then somewhere below in the proof tree we have $\mathcal{F}$ on the left side and $\mathcal{G}$ on the right side of the open branch. Our open branch will continue where by induction $\mathcal{F}$ is *true* and $\mathcal{G}$ is *false* therefore $\mathcal{F} \to \mathcal{G}$ is *false*.

In other words,

$$\mathcal{F} \Rightarrow 1$$
$$\mathcal{G} \Rightarrow 0$$
$$\therefore\ \mathcal{F} \to \mathcal{G} \Rightarrow 0$$

XI. *Negation on the left:*

Let $\neg\mathcal{F}$ be on the left side of the open branch, then somewhere below in the proof tree we have $\mathcal{F}$ on the right side of the open branch. Our open branch will continue where by induction $\mathcal{F}$ is *false* therefore $\neg\mathcal{F}$ is *true*.

XII. *Negation on the right:*

Let $\neg\mathcal{F}$ be on the right side of the open branch, then somewhere below in the proof tree we have $\mathcal{F}$ on the left side of the open branch. Our open branch will continue where by induction $\mathcal{F}$ is *true* therefore $\neg\mathcal{F}$ is *false*.

The result of this construction is an interpretation $\mathcal{I}$ that satisfies $\delta_0, \delta_1, \delta_2, \dots, \delta_{n-1}$ but not ψ. ■

For example, assume we have the following proof tree:

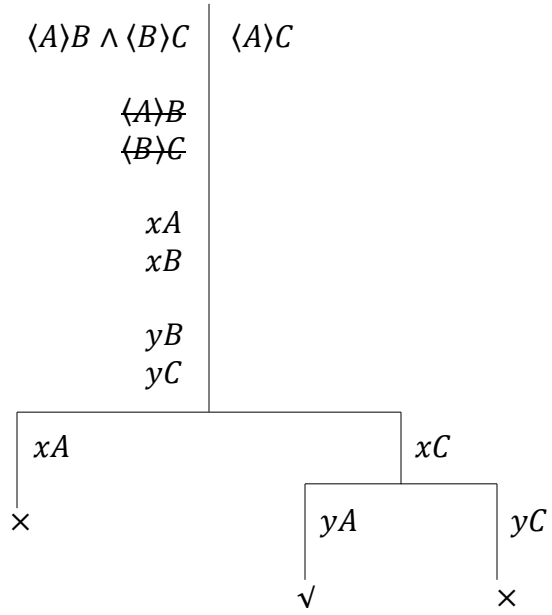

Figure 3.5: Completeness Proof Example

In this tree, we have one open branch with six atomic formulas, $xA, xB, yB, yC, xC,$ and yA.

For constructing $\mathcal{I}$, our universe of discourse contains two constants, $\{x, y\}$. Thus, our interpretation $\mathcal{I}$ will be as follows:

$$\mathcal{I}(A) = \{x\}, \mathcal{I}(B) = \{x, y\}, \mathcal{I}(C) = \{y\}$$

We have $\langle A\rangle B$, $\langle B\rangle C$ on the left and $\langle A\rangle C$ on the right along the open branch. We prove what appears in the left is *true* and what appears in the right is *false* as follows:

1. $\langle A\rangle B$ on the left:

$$xA \Rightarrow True$$
$$xB \Rightarrow True$$
$$\therefore \langle A\rangle B \Rightarrow True$$

2. $\langle B\rangle C$ on the left:

$$yB \Rightarrow True$$

$$yC \Rightarrow True$$

$$\therefore \langle B \rangle C \Rightarrow True$$

3. $\langle A \rangle C$ on the right:

$$xC \Rightarrow False \quad \text{or} \quad yA \Rightarrow False$$

$$\therefore \langle A \rangle C \Rightarrow 0$$

■

Chapter 4: The Soundness of the Monadic Hybrid Calculus $\mathcal{MHC}$

We continue presenting our proofs and in this chapter we present our tableau proof for the Soundness of the Monadic Hybrid Calculus $\mathcal{MHC}$ system. Prior to the proof, we give definitions and theorems that are necessary for following and understanding of our proof.

4.1. A Soundness Proof for the Monadic Hybrid Calculus $\mathcal{MHC}$

Definition 4.1.1 *(Soundness)*: Let δ be a set of axioms in the formal system $\mathcal{MHC}$. If ψ can be derived from δ using a tableau so that $\delta \vdash \psi$, then every interpretation $\mathcal{I}$ of δ is also an interpretation of ψ. In other words, if $\delta \vdash \psi$ then if $\mathcal{I} \vDash \delta$ we have $\mathcal{I} \vDash \psi$.

Theorem 4.1.2. *The tableau proof method is sound.*

Proof 4.1.3. Let $\delta_0, \delta_1, \delta_2, \dots, \delta_{n-1} \rightarrow \psi$ be an assertion whose tableau proof succeeds. We will show that for every interpretation $\mathcal{I}$, if $\mathcal{I} \vDash \delta$ then $\mathcal{I} \vDash \psi$.
Assume there is an $\mathcal{I}$ such that $\mathcal{I} \vDash \delta$ but $\mathcal{I} \nvDash \psi$. We will show that this assumption leads to a contradiction. We will say that a refutation appears at a node if there is an interpretation that makes true all the left hand formulas inherited at that point and makes false all the right hand formulas inherited at that point.

We start from the root of the proof tree traveling downward to a leaf. We will show that at every point of the proof tree where a refutation appears and there are rules enabled on the branch there is a node strictly below that also has a refutation. In each case we have a node n and an interpretation $\mathcal{I}$ that satisfies all the left hand formulas inherited at n and refutes all the right hand formulas inherited at n.

By our assumption there is a refutation at the root. We consider each of the tableau rules in turn and show that they propagate refutations as described:

I. $\langle \mathcal{P} \rangle \mathcal{Q}$ *as True:*

Suppose the path we are constructing has reached node n and the next rule applied involves a formula $\langle \mathcal{P} \rangle \mathcal{Q}$ further up the path on the left. Following the rule we must have put $v\mathcal{P}$ and $v\mathcal{Q}$, v a "fresh" constant, on the left. Call the node just below n', we extend our path to n' and show that we can associate a refutation $\mathcal{I}'$ with it. Since $\langle \mathcal{P} \rangle \mathcal{Q}$ is *true* there is an individual i in $\mathcal{I}(1)$ such that $\mathcal{I}, i \vDash \mathcal{P}$ and $\mathcal{I}, i \vDash \mathcal{Q}$. If we set the refutation $\mathcal{I}'$, which is located at node n' right below $v\mathcal{P}$ and $v\mathcal{Q}$, equal to $\mathcal{I}$ except that $\mathcal{I}'(v) = i$ then $\mathcal{I}' \vDash v\mathcal{P}$ and $\mathcal{I}' \vDash v\mathcal{Q}$ and since v does not occur above node n, then $\mathcal{I}'$ satisfies everything appearing on the left above n' and refutes everything on the right.

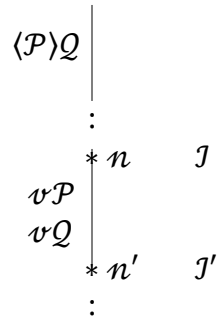

Figure 4.1: $\langle \mathcal{P} \rangle \mathcal{Q}$ on the *left*

II. $\langle \mathcal{P} \rangle \mathcal{Q}$ *as False:*

Suppose we have extended our path to node n with associated refutation $\mathcal{I}$ and the next rule applied involves a formula $\langle \mathcal{P} \rangle \mathcal{Q}$ on the right. Then for some constant p already used on the branch the tree splits below n into two branches, one with $p\mathcal{P}$ on the right and one with $p\mathcal{Q}$ on the right. Since $\mathcal{I}$ makes $\langle \mathcal{P} \rangle \mathcal{Q}$ false it must make at least one of $p\mathcal{P}$ and $p\mathcal{Q}$ *false*. Extend the path down one of the branches with a *false* formula down to the node n' just below the formula. If we associate $\mathcal{I}$ itself with n' it acts as a refutation.

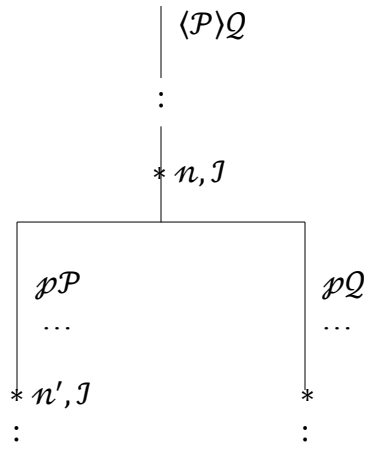

Figure 4.2: $\langle\mathcal{P}\rangle\mathcal{Q}$ on the *right*

Note: In rare case where no constants are already in use and the tableau rule of $\langle\mathcal{P}\rangle\mathcal{Q}$ as false selects a fresh constant $\mathcal{v}$ then we extend the path down to $\mathcal{n}'$ in both branch splits. Since $\langle\mathcal{P}\rangle\mathcal{Q}$ is *false* then there is an individual i in $\mathcal{I}(1)$ such that $\mathcal{I}, i \nvDash \mathcal{P}$ or $\mathcal{I}, i \nvDash \mathcal{Q}$. If we set the refutation $\mathcal{I}'$, which is located at node $\mathcal{n}'$ right below the split where $\mathcal{v}\mathcal{P}$ appears or the split where $\mathcal{v}\mathcal{Q}$ appears, equal to $\mathcal{I}$ except that $\mathcal{I}'(\mathcal{v}) = i$ then $\mathcal{I}' \nvDash \mathcal{v}\mathcal{P}$ or $\mathcal{I}' \nvDash \mathcal{v}\mathcal{Q}$. Since $\mathcal{v}$ does not occur above node $\mathcal{n}$ then $\mathcal{I}'$ satisfies everything appearing on the left above $\mathcal{n}'$ and refutes everything on the right.

III. $[\mathcal{P}]\mathcal{Q}$ *as True:*

Suppose we have extended our path to node $\mathcal{n}$ with associated refutation $\mathcal{I}$ and the next rule applied involves a formula $[\mathcal{P}]\mathcal{Q}$ on the left. Then for some constant $\mathcal{p}$ already used on the branch the tree splits below $\mathcal{n}$ into two branches, one with $\mathcal{p}\mathcal{P}$ on the right and one with $\mathcal{p}\mathcal{Q}$ on the left. Since $\mathcal{I}$ makes $[\mathcal{P}]\mathcal{Q}$ *true* it must make $\mathcal{p}\mathcal{P}$ *false* or $\mathcal{p}\mathcal{Q}$ *true*. Extend the path down the left if $\mathcal{p}\mathcal{P}$ is *false* or down the right if $\mathcal{p}\mathcal{Q}$ is *true* to the node $\mathcal{n}'$ just below the formula. If we associate $\mathcal{I}$ itself with $\mathcal{n}'$ it acts as a refutation.

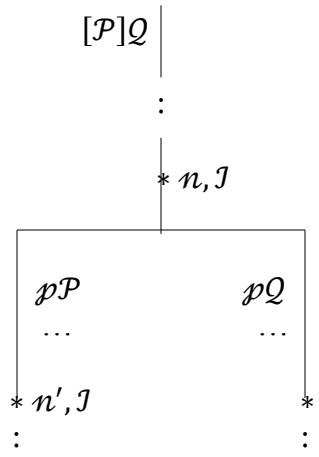

Figure 4.3: $[\mathcal{P}]\mathcal{Q}$ on the *left*

Note: Similar to the previous tableau rule, in the rare case where no constants are already in use and the tableau rule of $[\mathcal{P}]\mathcal{Q}$ as *true* selects a fresh constant v then we extend the path down to n' in both branch splits and since $[\mathcal{P}]\mathcal{Q}$ is *true* then there is an individual i in $\mathcal{I}(1)$ such that $\mathcal{I}, i \nvDash \mathcal{P}$ or $\mathcal{I}, i \vDash \mathcal{Q}$. If we set the refutation $\mathcal{I}'$, which is located at node n' right below the split where $v\mathcal{P}$ appears or the split where $v\mathcal{Q}$ appears, equal to $\mathcal{I}$ except that $\mathcal{I}'(v) = i$ then $\mathcal{I}' \nvDash v\mathcal{P}$ or $\mathcal{I}' \vDash v\mathcal{Q}$. Since v does not occur above node n then $\mathcal{I}'$ satisfies everything appearing on the left above n' and refutes everything on the right.

IV. $[\mathcal{P}]\mathcal{Q}$ *as False:*

Suppose the path we are constructing has reached node n and the next rule applied involved a formula $[\mathcal{P}]\mathcal{Q}$ further up the path on the right. Following the rule we must have put $v\mathcal{P}$, v a "fresh" constant, on the left and $v\mathcal{Q}$ on the right. Call the node just below n', we extend our path to n' and show that we can associate a refutation $\mathcal{I}'$ with it. Since $[\mathcal{P}]\mathcal{Q}$ is *false* there is an individual i in $\mathcal{I}(1)$ such that $\mathcal{I}, i \vDash \mathcal{P}$ and $\mathcal{I}, i \nvDash \mathcal{Q}$. If we set the refutation $\mathcal{I}'$, which is located at node n' right below $v\mathcal{P}$ and $v\mathcal{Q}$, equal to $\mathcal{I}$ except that $\mathcal{I}'(v) = i$ then $\mathcal{I}' \vDash v\mathcal{P}$ and $\mathcal{I}' \nvDash v\mathcal{Q}$ and since v does not occur above node n, then $\mathcal{I}'$ satisfies everything appearing on the left above n' and refutes everything on the right.

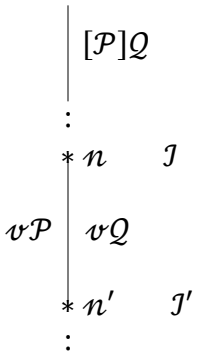

Figure 4.4: $[\mathcal{P}]\mathcal{Q}$ on the *right*

V. *Conjunction as True:*

Suppose the path we are constructing ends in a node $\mathcal{n}$ with a refutation $\mathcal{I}$ and the next rule applied involves a formula $\mathcal{F} \wedge \mathcal{G}$ on the left. Then just below it we will find $\mathcal{F}$ and $\mathcal{G}$ on the left and a node n' below them. Extend the path to n' and associate $\mathcal{I}$ with $\mathcal{n}'$. Since $\mathcal{I} \vDash \mathcal{F} \wedge \mathcal{G}$, $\mathcal{I} \vDash \mathcal{F}$ and $\mathcal{I} \vDash \mathcal{G}$ and $\mathcal{I}$ itself is a refutation.

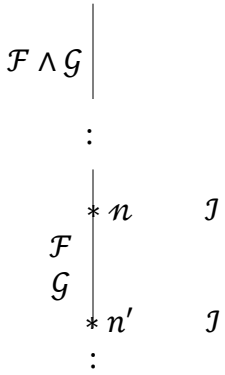

Figure 4.5: Conjunction on the *left*

VI. *Conjunction as False:*

Suppose the path we are constructing ends in $\mathcal{n}$ and $\mathcal{I}$ and the next rule involves $\mathcal{F} \wedge \mathcal{G}$ on the right. Then the tree splits into two branches one with $\mathcal{F}$ on the right and one with $\mathcal{G}$ on the right. Since $\mathcal{I} \nvDash \mathcal{F} \wedge \mathcal{G}$, $\mathcal{I}$ must make at least one of $\mathcal{F}$ and $\mathcal{G}$ *false*. Extend the path down one of these *false* branches to the node n' just below

the *false* formula. Since $\mathcal{I}$ refutes the formula just above n', $\mathcal{I}$ can serve as the refutation associated with n'.

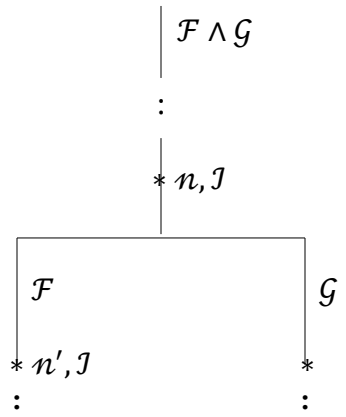

Figure 4.6: Conjunction on the *right*

VII. *Disjunction as True:*

Suppose the path we are constructing ends in n and $\mathcal{I}$ and the next rule involves $\mathcal{F} \vee \mathcal{G}$ on the left. Then the tree splits into two branches one with $\mathcal{F}$ on the left and one with $\mathcal{G}$ on the left. Since $\mathcal{I} \vDash \mathcal{F} \vee \mathcal{G}$, $\mathcal{I}$ must make at least one of $\mathcal{F}$ and $\mathcal{G}$ *true*. Extend the path down one of these *true* branches to the node n' just below the *true* formula. Since $\mathcal{I}$ satisfies the formula just above n', $\mathcal{I}$ can serve as the refutation associated with n'.

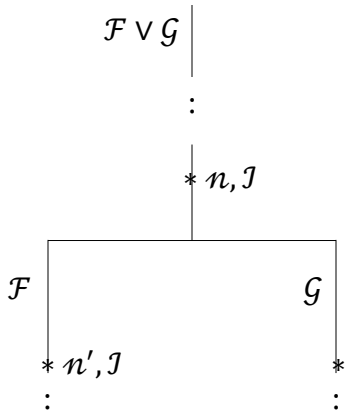

Figure 4.7: Disjunction on the *left*

VIII. *Disjunction as False:*

Suppose the path we are constructing ends in a node n with a refutation $\mathcal{I}$ and the next rule applied involves a formula $\mathcal{F} \vee \mathcal{G}$ on the right. Then just below it we will find $\mathcal{F}$ and $\mathcal{G}$ and a node n' below them. Extend the path to n' and associate $\mathcal{I}$ with n. Since $\mathcal{I} \not\models \mathcal{F} \vee \mathcal{G}$, $\mathcal{I} \not\models \mathcal{F}$ and $\mathcal{I} \not\models \mathcal{G}$ and $\mathcal{I}$ itself is a refutation.

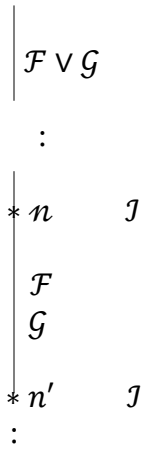

Figure 4.8: Disjunction on the *right*

IX. *Implication as True:*

Suppose the path we are constructing ends in n and $\mathcal{I}$ and the next rule involves $\mathcal{F} \rightarrow \mathcal{G}$ on the left. Then the tree splits into two branches one with $\mathcal{F}$ on the right and one with $\mathcal{G}$ on the left. Since $\mathcal{I} \models \mathcal{F} \rightarrow \mathcal{G}$, $\mathcal{I}$ must make $\mathcal{F}$ *false* or $\mathcal{G}$ *true*. Extend the path down the left branch if $\mathcal{I} \not\models \mathcal{F}$ or down the right branch if $\mathcal{I} \models \mathcal{G}$ to the node n' just below the formula. Since $\mathcal{I}$ satisfies or refutes, as appropriate, the formula just above n', $\mathcal{I}$ can serve as the refutation associated with n'.

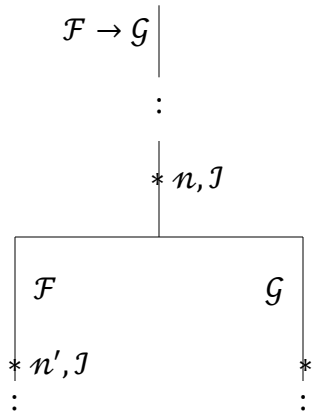

Figure 4.9: Implication on the *left*

X. *Implication as False:*

Suppose the path we are constructing ends in a node n with a refutation $\mathcal{I}$ and the next rule applied involves a formula $\mathcal{F} \to \mathcal{G}$ on the right. Then just below it we will find $\mathcal{F}$ and $\mathcal{G}$ and a node n' below them. Extend the path to n' and associate $\mathcal{I}$ with n. Since $\mathcal{I} \nvDash \mathcal{F} \to \mathcal{G}$, $\mathcal{I} \vDash \mathcal{F}$ and $\mathcal{I} \nvDash \mathcal{G}$ and $\mathcal{I}$ is a refutation.

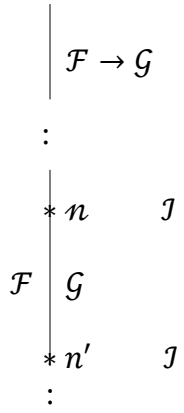

Figure 4.10: Implication on the *right*

XI. *Negation as True:*

Suppose the path we are constructing ends in a node n with a refutation $\mathcal{I}$ and the next rule applied involves a formula $\neg\mathcal{F}$ on the left. Then just below it we will

find $\mathcal{F}$ with a node n' below it. Extend the path to n' and associate $\mathcal{I}$ with n. Since $\mathcal{I} \vDash \neg\mathcal{F}$, $\mathcal{I} \nvDash \mathcal{F}$ and $\mathcal{I}$ itself is a refutation.

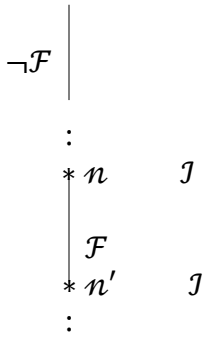

Figure 4.11: Negation on the *left*

XII. *Negation as False:*

Suppose the path we are constructing ends in a node n with a refutation $\mathcal{I}$ and the next rule applied involves a formula $\neg\mathcal{F}$ on the right. Then just below it we will find $\mathcal{F}$ with a node n' below it. Extend the path to n' and associate $\mathcal{I}$ with n. Since $\mathcal{I} \nvDash \neg\mathcal{F}$, $\mathcal{I} \vDash \mathcal{F}$ and $\mathcal{I}$ itself is a refutation.

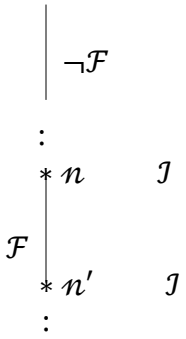

Figure 4.12: Negation on the *right*

We have shown that every node with a refutation attached to it and a rule enabled on the path leading to the node has a node below it with a refutation attached to it. Since all paths are closed, this is a contradiction.

■

Chapter 5: Monadic Hybrid Calculus $\mathcal{MHC}$ Conversion Algorithms

In this chapter, we introduce the syntax and semantics of the Extended Monadic Hybrid Calculus $\mathcal{MHC}^+$ system that will be used within the conversion process from and to the Monadic Hybrid Calculus $\mathcal{MHC}$ system. We then give our algorithm to convert formulas expressed in the Monadic Predicate Calculus $\mathcal{MPC}$ to logically equivalent forms of Monadic Hybrid Calculus $\mathcal{MHC}$ formulas. We also introduce a second algorithm to convert Monadic Hybrid Calculus $\mathcal{MHC}$ formulas to their logically equivalent forms of Monadic Predicate Calculus $\mathcal{MPC}$ formulas. Each algorithm is followed by an illustration section that explains the methodology of this algorithm along with examples of formulas being transformed from one system to another using said algorithm. Moreover, we present examples of natural language statements and their formulas taken from different textbooks in order to show how to translate directly from the $\mathcal{MPC}$ system to the $\mathcal{MHC}$ system. Since the $\mathcal{MHC}$ is logically equivalent to the $\mathcal{MPC}$ system then we present our model theoretic proof of the Compactness of the Monadic Hybrid Calculus $\mathcal{MHC}$ at the end of this chapter.

Incidentally, the equivalence result does not automatically give us Completeness and Soundness. For example, suppose that every interpretation that makes δ true makes ψ true. The Completeness means that there must be a proof of $\delta \to \psi$. We can translate δ and ψ into $\mathcal{MPC}$ formulas δ' and ψ' and Completeness (with respect to some $\mathcal{MPC}$ proof system) gives us a proof of ψ' from δ' but this is an $\mathcal{MPC}$ proof, not an $\mathcal{MHC}$ proof. We have to be able to translate $\mathcal{MPC}$ proofs back into $\mathcal{MHC}$ proofs and this is not straightforward approach.

5.1. The Syntax of the Extended Monadic Hybrid Calculus $\mathcal{MHC}^+$

The Extended Monadic Hybrid Calculus $\mathcal{MHC}^+$ uses the following symbols:

I. An infinite set of upper case letters $\{A, B, C, \dots A_1, B_1, C_1, \dots\}$ representing properties;

II. An infinite set of lower case letters $\{x, y, z, \ldots\ x_1, y_1, z_1, \ldots\}$ representing individuals;

III. Logical symbols: $\wedge$ for *conjunction*, $\vee$ for *disjunction*, $\neg$ for *negation*, $\rightarrow$ for *implication*;

IV. Square brackets around a property formula, $[\mathcal{P}]$, for *universal quantification* and angle brackets around a property formula, $\langle\mathcal{P}\rangle$, for *existential quantification*.

V. The *universal quantifier*, $\forall$, and the *existential quantifier*, $\exists$, for binding individuals.

VI. Parentheses ( ) to combine two or more formulas.

Expressions in $\mathcal{MHC}^+$ fall in two categories, *property formulas* and *absolute formulas.*

Definition 5.1.1. *An Extended Monadic Hybrid Calculus property formula is a Boolean combination of property variables. In other words, either a property variable or an expression of the form:* $(\mathcal{P} \wedge \mathcal{Q}), (\mathcal{P} \vee \mathcal{Q}), (\mathcal{P} \rightarrow \mathcal{Q}), \neg\mathcal{P}, 0$ *or* 1*, where* $\mathcal{P}$ *and* $\mathcal{Q}$ *are property formulas.*

Definition 5.1.2. An Extended Monadic Hybrid Calculus absolute formula is either:

i. $\mathcal{v}\mathcal{P}$, where $\mathcal{P}$ is a property formula and $\mathcal{v}$ is a constant. Such as: xA *"x is an apple",*

ii. $[\mathcal{A}]\mathcal{P}$, where $\mathcal{A}$ and $\mathcal{P}$ are both property formulas with *universal* quantification for the first property. Such as: $[S](P \wedge F)$ *"Anything that is a Strawberry is a Plant and a Fruit"*,

iii. $\langle\mathcal{A}\rangle\mathcal{P}$, where $\mathcal{A}$ and $\mathcal{P}$ are both property formulas with *existential* quantification for the first property. Such as: $\langle O\rangle(P \wedge F)$ *"Some things that are Organic are Plants and Fruit"*,

iv. $\forall\mathcal{v}\mathcal{F}$, where $\mathcal{F}$ is a formula and $\mathcal{v}$ is a constant,

v. $\exists\mathcal{v}\mathcal{F}$, where $\mathcal{F}$ is a formula and $\mathcal{v}$ is a constant,

vi. A Boolean combination using logical symbols, namely $\wedge$ for *conjunction*, $\vee$ for *disjunction* or $\rightarrow$ for *implication*, to connect two or more formulas. Such that if $\mathcal{G}$

and $\mathcal{H}$ are formulas so are $(\mathcal{G} \wedge \mathcal{H}), (\mathcal{G} \vee \mathcal{H}), (\mathcal{G} \rightarrow \mathcal{H})$.

vii. Boolean negation $\neg$, so that if $\mathcal{F}$ is a formula so is $\neg\mathcal{F}$. For instance, $\neg[A]R$, *"Not all Apples are Red",*

viii. Truth constants: 0 for *False* and 1 for *True.*

5.2. The Semantics of the Extended Monadic Hybrid Calculus $\mathcal{MHC}^+$

Definition 5.2.1. An interpretation $\mathcal{I}$ is a function that assigns to 1 a nonempty domain of discourse $\mathcal{I}(1)$-a set of individuals (people, things, days, …etc.), that assigns to each property variable $\mathcal{V}$ a set $\mathcal{I}(\mathcal{V})$ of individuals and that assigns to each individual variable $\mathcal{v}$ an element $\mathcal{I}(\mathcal{v})$ of $\mathcal{I}(1)$.

Definition 5.2.2. The semantics of $\mathcal{MHC}^+$ formulas is given by the following rules (for all $\mathcal{MHC}^+$ formulas, $\mathcal{F}$ and $\mathcal{G}$, for all $\mathcal{MHC}^+$ property formulas, $\mathcal{P}$ and $\mathcal{Q}$, and $\mathcal{I}[\mathcal{v}/a] \vDash \mathcal{F}$ is the interpretation identical to $\mathcal{I}$ except that $\mathcal{I}(\mathcal{v}) = a$):

$\mathcal{I} \vDash \mathcal{v}\mathcal{P}$	*iff*	$\mathcal{I}, \mathcal{I}(\mathcal{v}) \vDash \mathcal{P}$
$\mathcal{I} \vDash [\mathcal{P}]\mathcal{Q}$	*iff*	for all a in $\mathcal{I}(1)$, if $\mathcal{I}, a \vDash \mathcal{P}$ then $\mathcal{I}, a \vDash \mathcal{Q}$
$\mathcal{I} \vDash \langle\mathcal{P}\rangle\mathcal{Q}$	*iff*	for some a in $\mathcal{I}(1)$, $\mathcal{I}, a \vDash \mathcal{P}$ and $\mathcal{I}, a \vDash \mathcal{Q}$
$\mathcal{I} \vDash \forall\mathcal{v}\mathcal{F}$	*iff*	for all a, $\mathcal{I}[\mathcal{v}/a] \vDash \mathcal{F}$
$\mathcal{I} \vDash \exists\mathcal{v}\mathcal{F}$	*iff*	for some a, $\mathcal{I}[\mathcal{v}/a] \vDash \mathcal{F}$
$\mathcal{I} \vDash \mathcal{F} \wedge \mathcal{G}$	*iff*	$\mathcal{I} \vDash \mathcal{F}$ and $\mathcal{I} \vDash \mathcal{G}$
$\mathcal{I} \vDash \mathcal{F} \vee \mathcal{G}$	*iff*	$\mathcal{I} \vDash \mathcal{F}$ and/or $\mathcal{I} \vDash \mathcal{G}$
$\mathcal{I} \vDash \mathcal{F} \rightarrow \mathcal{G}$	*iff*	It is not the case that: $\mathcal{I} \vDash \mathcal{F}$ and not $\mathcal{I} \vDash \mathcal{G}$
$\mathcal{I} \vDash \neg\mathcal{F}$	*iff*	$\mathcal{I} \nvDash \mathcal{F}$
$\mathcal{I} \vDash 1$		
$\mathcal{I} \nvDash 0$		

Definition 5.2.3. The semantics of $\mathcal{MHC}^+$ property formulas is given by the following rules (for any $\mathcal{MHC}^+$ property formulas, $\mathcal{P}$ and $\mathcal{Q}$, any element a of $\mathcal{I}(1)$, and any property variable $\mathcal{V}$):

$\mathcal{I}, a \vDash (\mathcal{P} \wedge \mathcal{Q})$	*iff*	$\mathcal{I}, a \vDash \mathcal{P}$ and $\mathcal{I}, a \vDash \mathcal{Q}$

$\mathcal{I}, a \vDash (\mathcal{P} \vee \mathcal{Q})$	*iff*	$\mathcal{I}, a \vDash \mathcal{P}$ and/or $\mathcal{I}, a \vDash \mathcal{Q}$
$\mathcal{I}, a \vDash \mathcal{P} \rightarrow \mathcal{Q}$	*iff*	It is not the case that $\mathcal{I}, a \vDash \mathcal{P}$ and not $\mathcal{I}, a \vDash \mathcal{Q}$
$\mathcal{I}, a \vDash \neg\mathcal{P}$	*iff*	$\mathcal{I}, a \nvDash \mathcal{P}$
$\mathcal{I}, a \vDash \mathcal{V}$	*iff*	$a \in \mathcal{I}(\mathcal{V})$
$\mathcal{I}, a \vDash 1$		
$\mathcal{I}, a \nvDash 0$		

Theorem 5.2.4. *Every Monadic Predicate Calculus formula is equivalent to one in the Monadic Hybrid Calculus.*

5.3. The Monadic Predicate Calculus $\mathcal{MPC}$ to the Monadic Hybrid Calculus $\mathcal{MHC}$ Conversion Algorithm

Algorithm 1: $\mathcal{MPC}$ to $\mathcal{MHC}$ Conversion

Input: A Monadic Predicate Calculus $\mathcal{MPC}$ formula

Output: A Monadic Hybrid Calculus $\mathcal{MHC}$ formula

1. Convert the formula into its logically equivalent Prenex formula of the form $\gamma\mathcal{P}$ where γ is a sequence of quantifiers and $\mathcal{P}$ is a quantifier-free $\mathcal{MPC}$ formula, thus it is also in $\mathcal{MHC}$ form while $\gamma\mathcal{P}$ itself is in $\mathcal{MHC}^+$ form.
2. If the immediate prefixed quantifier is an existential quantifier, such as $\exists x\mathcal{P}$, then:
3. Convert $\mathcal{P}$ into its Disjunctive Normal Form (DNF), so that the result is $\exists x((\mathcal{Q}_0 \wedge \mathcal{Q}_1 \wedge \mathcal{Q}_2 \wedge \ldots \wedge \mathcal{Q}_m) \vee (\mathcal{Q}_{m+1} \wedge \mathcal{Q}_{m+2} \wedge \mathcal{Q}_{m+3} \ldots) \vee \ldots)$.
4. Distribute the existential quantifier $\exists x$ on all clauses, so that the result is $\exists x(\mathcal{Q}_0 \wedge \mathcal{Q}_1 \wedge \mathcal{Q}_2 \wedge \ldots \wedge \mathcal{Q}_m) \vee \exists x(\mathcal{Q}_{m+1} \wedge \mathcal{Q}_{m+2} \wedge \mathcal{Q}_{m+3} \ldots) \vee \ldots)$.
5. In each clause, arrange all atomic formulas in which x occurs in front, yielding $\exists x(x\mathcal{R}_0 \wedge x\mathcal{R}_1 \wedge x\mathcal{R}_2 \wedge \ldots \wedge \mathcal{S}_0 \wedge \mathcal{S}_1 \wedge \mathcal{S}_2 \wedge \ldots)$ to form one clause as $\exists x((x\mathcal{R}_0 \wedge x\mathcal{R}_1 \wedge x\mathcal{R}_2 \wedge \ldots) \wedge \ldots \wedge \mathcal{S}_0 \wedge \mathcal{S}_1 \wedge \mathcal{S}_2 \wedge \ldots)$.
6. For each clause formed in the previous step, take the bound variable as a common factor outside its clause, as in $\exists x((x\mathcal{R}_0 \wedge x\mathcal{R}_1 \wedge x\mathcal{R}_2 \wedge \ldots) \wedge \ldots \wedge \mathcal{S}_0 \wedge \mathcal{S}_1 \wedge \mathcal{S}_2 \wedge \ldots)$ then $\exists x(x(\mathcal{R}_0 \wedge \mathcal{R}_1 \wedge \mathcal{R}_2 \wedge \ldots) \wedge \ldots \wedge \mathcal{S}_0 \wedge \mathcal{S}_1 \wedge \mathcal{S}_2 \wedge \ldots)$.
7. Push the quantifier inside the bigger clause next to the common factor (the bound variable), from $\exists x(x(\mathcal{R}_0 \wedge \mathcal{R}_1 \wedge \mathcal{R}_2 \wedge \ldots) \wedge \ldots \wedge \mathcal{S}_0 \wedge \mathcal{S}_1 \wedge \mathcal{S}_2 \wedge \ldots)$ we get $(\exists x\, x(\mathcal{R}_0 \wedge \mathcal{R}_1 \wedge \mathcal{R}_2 \wedge \ldots) \wedge \ldots \wedge \mathcal{S}_0 \wedge \mathcal{S}_1 \wedge \mathcal{S}_2 \wedge \ldots)$ since x does not occur in any of $\mathcal{S}_i$.
8. Change the quantifier along with the common factor (the bound variable) to its equivalent form of $\mathcal{MHC}$ quantifier, as in:
   $(\exists x\, x(\mathcal{R}_0 \wedge \mathcal{R}_1 \wedge \ldots) \wedge \ldots \wedge \mathcal{S}_0 \wedge \mathcal{S}_1 \wedge \ldots) \equiv (\langle 1\rangle(\mathcal{R}_0 \wedge \mathcal{R}_1 \wedge \ldots) \wedge \ldots \wedge \mathcal{S}_0 \wedge \mathcal{S}_1 \wedge \ldots)$
9. If the immediate prefixed quantifier is a universal quantifier, $\forall x\mathcal{P}$, then:

10. Convert $\mathcal{P}$ into its Conjunctive Normal Form (CNF), so that the result is $\forall x((\mathcal{Q}_0 \vee \mathcal{Q}_1 \vee \mathcal{Q}_2 \vee \ldots \vee \mathcal{Q}_m) \wedge (\mathcal{Q}_{m+1} \vee \mathcal{Q}_{m+2} \vee \mathcal{Q}_{m+3} \ldots) \wedge \ldots)$.
11. Distribute the universal quantifier $\forall x$ on all clauses, so that the result is $\forall x(\mathcal{Q}_0 \vee \mathcal{Q}_1 \vee \mathcal{Q}_2 \vee \ldots \vee \mathcal{Q}_m) \wedge \forall x(\mathcal{Q}_{m+1} \vee \mathcal{Q}_{m+2} \vee \mathcal{Q}_{m+3} \ldots) \wedge \ldots)$.
12. In each clause, arrange all atomic formulas in which x occurs in front, yielding $\forall x(x\mathcal{R}_0 \vee x\mathcal{R}_1 \vee x\mathcal{R}_2 \vee \ldots \vee \mathcal{S}_0 \vee \mathcal{S}_1 \vee \mathcal{S}_2 \vee \ldots)$ to form one clause as $\forall x((x\mathcal{R}_0 \vee x\mathcal{R}_1 \vee x\mathcal{R}_2 \vee \ldots) \vee \ldots \vee \mathcal{S}_0 \vee \mathcal{S}_1 \vee \mathcal{S}_2 \vee \ldots)$.
13. For each clause formed in the previous step, take the bound variable as a common factor outside its clause, as in $\forall x((x\mathcal{R}_0 \vee x\mathcal{R}_1 \vee x\mathcal{R}_2 \vee \ldots) \vee \ldots \vee \mathcal{S}_0 \vee \mathcal{S}_1 \vee \mathcal{S}_2 \vee \ldots)$ then $\forall x(x(\mathcal{R}_0 \vee \mathcal{R}_1 \vee \mathcal{R}_2 \vee \ldots) \vee \ldots \vee \mathcal{S}_0 \vee \mathcal{S}_1 \vee \mathcal{S}_2 \vee \ldots)$.
14. Push the quantifier inside the bigger clause next to the common factor (the bound variable), from $\forall x(x(\mathcal{R}_0 \vee \mathcal{R}_1 \vee \mathcal{R}_2 \vee \ldots) \vee \ldots \vee \mathcal{S}_0 \vee \mathcal{S}_1 \vee \mathcal{S}_2 \vee \ldots)$ we get $(\forall x\, x(\mathcal{R}_0 \vee \mathcal{R}_1 \vee \mathcal{R}_2 \vee \ldots) \vee \ldots \vee \mathcal{S}_0 \vee \mathcal{S}_1 \vee \mathcal{S}_2 \vee \ldots)$ since x does not occur in any of $\mathcal{S}_i$.
15. Change the quantifier along with the common factor (the bound variable) to its equal form of $\mathcal{MHC}$ quantifier, such as:
    $(\forall x\, x(\mathcal{R}_0 \vee \mathcal{R}_1 \vee \ldots) \vee \ldots \vee \mathcal{S}_0 \vee \mathcal{S}_1 \vee \ldots) \equiv ([1](\mathcal{R}_0 \vee \mathcal{R}_1 \vee \ldots) \vee \ldots \vee \mathcal{S}_0 \vee \mathcal{S}_1 \vee \ldots)$
16. If still there exists at least one quantifier then eliminate the next immediate prefixed quantifier by iterating the algorithm on the formula that results from the last iteration until we have the last formula $\gamma_0\mathcal{P}_0$ where γ_0 is an empty sequence of quantifiers and $\mathcal{P}_0$ is a $\mathcal{MHC}$ formula.

Theorem 5.3.1. *For any Monadic Predicate Calculus formula α: α is logically equivalent to its Monadic Hybrid Calculus translation.*

Proof 5.3.2. Most of the translations involves procedures (like converting to DNF/CNF) that are known to be equivalence preserving. The only nonstandard transformations are changing $\exists x\, x\mathcal{P}$ to $\langle 1\rangle\mathcal{P}$ and $\forall x\, x\mathcal{P}$ to $[1]\mathcal{P}$. We will prove the first in details .

For the existential case, we have to prove that if $\mathcal{I} \vDash \exists x\, x\mathcal{P}$ then $\mathcal{I} \vDash \langle 1\rangle\mathcal{P}$ and if $\mathcal{I} \vDash \langle 1\rangle\mathcal{P}$ then $\mathcal{I} \vDash \exists x\, x\mathcal{P}$. Suppose first that $\mathcal{I} \vDash \exists x\, x\mathcal{P}$. Then, for some i, $\mathcal{I}[x/i] \vDash x\mathcal{P}$.

This means $\mathcal{I}[x/i], \mathcal{I}[x/i](x) \vDash \mathcal{P}$. Since $\mathcal{I}[x/i](x) \vDash i$, we have $\mathcal{I}[x/i], i \vDash \mathcal{P}$. But $\mathcal{P}$ is a property formula and has no constants, in particular no occurrence of x. Thus $\mathcal{I}, i \vDash \mathcal{P}$. Finally, since $\mathcal{I}, i \vDash 1$, we have $\mathcal{I} \vDash \langle 1 \rangle \mathcal{P}$.

In the other direction, assume $\mathcal{I} \vDash \langle 1 \rangle \mathcal{P}$. Then, $\mathcal{I}, i \vDash \mathcal{P}$ for some i. This means $\mathcal{I}[x/i], i \vDash \mathcal{P}$ so $\mathcal{I}[x/i] \vDash x\mathcal{P}$ and hence $\mathcal{I} \vDash \exists x\, x\mathcal{P}$.

The proof of the equivalence of $\forall x\, x\mathcal{P}$ and $[1]\mathcal{P}$ is similar.

5.4. $\mathcal{MPC}$ to $\mathcal{MHC}$ Conversion Algorithm Illustration

A full transformation of a Monadic Predicate Calculus formula to its logically equivalent form of Monadic Hybrid Calculus formula is an indirect process in most cases. A formula has to go through iterations in the $\mathcal{MPC} - \mathcal{MHC}$'s algorithm in order to be completely transformed. In between these iterations, intermediate formulas produced are often in $\mathcal{MHC}^+$ syntax that allows for both the Monadic Predicate Calculus syntax and the Monadic Hybrid Calculus syntax to exist simultaneously. Thus, it was imperative that we introduce the syntax and semantics of the $\mathcal{MHC}^+$ prior to the construction of the algorithm.

In step 1 of the $\mathcal{MPC}$ to $\mathcal{MHC}$ conversion algorithm, we transfer the formula into a logically equivalent Prenex formula. The purpose of this step is to allow us to eliminate one quantifier per algorithm iteration. In such a way, we can facilitate the insertion of each quantifier into its corresponding clause where the bound variable occurs without distorting the original interpretation of the entire formula.

Since we assume that the formula could be a result of the previous iteration of the algorithm then we assume that there is a possibility that it has mixed notations of $\mathcal{MPC}$ and $\mathcal{MHC}$ that is $\mathcal{MHC}^+$ syntax.

In step 2 of the algorithm, we have to enquire about whether the quantifier is *existential* or *universal* for our further steps of the algorithm since they depend on the sort of the quantifier being eliminated. If we are eliminating an *existential* quantifier then, in step 3, we transform the formula $\mathcal{P}_n$ to its Disjunctive Normal Form (DNF). The purpose of this conversion is because, in a later step, we will need to distribute the quantifier

being eliminated on all clauses and we are only allowed to do so if and only if the quantifier is *existential*.

In step 4 of the algorithm, we distribute the *existential* quantifier, $\exists x$, on every disjunctive clause where the variable x occurs at least once thus we facilitate the insertion of the quantifier into such a clause in a later step. We are only allowed to distribute an *existential* quantifier over disjunctions thus we distribute $\exists x$ as follow, if $\exists x((\mathcal{Q}_0 \wedge \mathcal{Q}_1 \wedge \mathcal{Q}_2 \wedge \ldots \wedge \mathcal{Q}_m) \vee (\mathcal{Q}_{m+1} \wedge \mathcal{Q}_{m+2} \wedge \mathcal{Q}_{m+3} \ldots) \vee \ldots)$ then $(\exists x(\mathcal{Q}_0 \wedge \mathcal{Q}_1 \wedge \mathcal{Q}_2 \wedge \ldots \wedge \mathcal{Q}_m) \vee \exists x(\mathcal{Q}_{m+1} \wedge \mathcal{Q}_{m+2} \wedge \mathcal{Q}_{m+3} \ldots) \vee \ldots)$.

In step 5, we rearrange the atomic formulas inside each conjunctive clause and form inner clauses that contain the atomic formulas where the variable x occurs. If, for example, $\exists x(x\mathcal{R}_0 \wedge \mathcal{S}_0 \wedge \mathcal{S}_1 \wedge x\mathcal{R}_1 \wedge x\mathcal{R}_2 \wedge \mathcal{S}_2 \wedge \ldots)$ then we rearrange the formula into $\exists x((x\mathcal{R}_0 \wedge x\mathcal{R}_1 \wedge x\mathcal{R}_2) \wedge \ldots \wedge \mathcal{S}_0 \wedge \mathcal{S}_1 \wedge \mathcal{S}_2 \wedge \ldots)$. Since we are dealing with one bound variable, x, then we will have only one inner clause that is joined with other atomic formulas. In this inner clause, as step 6 of the algorithm indicates, we take the variable x as a common factor thus the inner clause $\exists x((x\mathcal{R}_0 \wedge x\mathcal{R}_1 \wedge x\mathcal{R}_2) \wedge \ldots \wedge \mathcal{S}_0 \wedge \mathcal{S}_1 \wedge \mathcal{S}_2 \wedge \ldots)$ will become $\exists x(x(\mathcal{R}_0 \wedge \mathcal{R}_1 \wedge \mathcal{R}_2) \wedge \ldots \wedge \mathcal{S}_0 \wedge \mathcal{S}_1 \wedge \mathcal{S}_2 \wedge \ldots)$. Here we can distribute the *existential* quantifier, $\exists x$, over the conjunctive inner clause since x does not occur in any of the formulas $\mathcal{S}_i$.

In step 7, we push the quantifier $\exists x$ right next to the clause where x has been taken as a common factor thus, in step 8, we merge the two and change them to the equivalent form of $\mathcal{MHC}$ quantifier so that $\ldots(\exists x\, x(\mathcal{R}_0 \wedge \mathcal{R}_1 \wedge \mathcal{R}_2) \wedge \ldots \wedge \mathcal{S}_0 \wedge \mathcal{S}_1 \wedge \mathcal{S}_2 \wedge \ldots)$ become $\ldots(\langle 1 \rangle(\mathcal{R}_0 \wedge \mathcal{R}_1 \wedge \mathcal{R}_2) \wedge \ldots \wedge \mathcal{S}_0 \wedge \mathcal{S}_1 \wedge \mathcal{S}_2 \wedge \ldots)$.

Step 9 of the algorithm detours the flow of conversion process. If the quantifier to be eliminated is *universal* then we transform $\mathcal{P}_n$ into its Conjunctive Normal Form (CNF), as in step 10, so we can distribute the quantifier on all related clauses. The CNF is conjunctions of clauses of disjunctions of $\mathcal{P}_n$'s atomic formulas. We convert into CNF because we will, in a later step, distribute the *universal* quantifier over the formulas and we are allowed to do so if and only if, unlike in the *existential* quantifier, we have conjunctive formulas.

In step 11 of the algorithm, we distribute the *universal* quantifier, $\forall x$, on every conjunctive clause where the variable x occurs at least once thus we facilitate the

insertion of the quantifier into such a clause in a further step. If, for example, we have $\forall x((\mathcal{Q}_0 \vee \mathcal{Q}_1 \vee \mathcal{Q}_2 \vee \ldots \vee \mathcal{Q}_m) \wedge (\mathcal{Q}_{m+1} \vee \mathcal{Q}_{m+2} \vee \mathcal{Q}_{m+3} \ldots) \wedge \ldots)$ then $\forall x$ will be distributed as $(\forall x(\mathcal{Q}_0 \vee \mathcal{Q}_1 \vee \mathcal{Q}_2 \vee \ldots \vee \mathcal{Q}_m) \wedge \forall x(\mathcal{Q}_{m+1} \vee \mathcal{Q}_{m+2} \vee \mathcal{Q}_{m+3} \ldots) \wedge \ldots)$.

In step 12, we rearrange the atomic formulas, inside each clause, and form inner clauses that contain the atomic formulas where the variable x occurs in. Thus, if $\forall x(x\mathcal{R}_0 \vee \mathcal{S}_0 \vee \mathcal{S}_1 \vee x\mathcal{R}_1 \vee x\mathcal{R}_2 \vee \mathcal{S}_2 \vee \ldots)$ then we rearrange the formula so $\forall x((x\mathcal{R}_0 \vee x\mathcal{R}_1 \vee x\mathcal{R}_2) \vee \ldots \vee \mathcal{S}_0 \vee \mathcal{S}_1 \vee \mathcal{S}_2 \vee \ldots)$. Since we are eliminating one quantifier and thus one variable then for each bigger clause we will form only one inner clause since all the atomic formulas where variable x occurs in will be group there.

In step 13 of the algorithm, we take the variable x in each inner clause formed in the previous step as a common factor. So, x in the formula $\forall x((x\mathcal{R}_0 \vee x\mathcal{R}_1 \vee x\mathcal{R}_2) \vee \ldots \vee \mathcal{S}_0 \vee \mathcal{S}_1 \vee \mathcal{S}_2 \vee \ldots)$ will become the common factor in the formula $\forall x(x(\mathcal{R}_0 \vee \mathcal{R}_1 \vee \mathcal{R}_2) \vee \ldots \vee \mathcal{S}_0 \vee \mathcal{S}_1 \vee \mathcal{S}_2 \vee \ldots)$. Hence, we are now able to push the quantifier $\forall x$, in step 14 of the algorithm, next to the common factor x and we are allowed to do so, over disjunctive formulas, since x does not occur in any of the formulas $\mathcal{S}_i$.

In step 15, we merge the quantifier $\forall x$ with the common factor x and we transform them into the equivalent form of $\mathcal{MHC}$ quantifier. Therefore, if $(\forall x\, x(\mathcal{R}_0 \vee \mathcal{R}_1 \vee \mathcal{R}_2) \vee \ldots \vee \mathcal{S}_0 \vee \mathcal{S}_1 \vee \mathcal{S}_2 \vee \ldots)$ then $([1](\mathcal{R}_0 \vee \mathcal{R}_1 \vee \mathcal{R}_2) \vee \ldots \vee \mathcal{S}_0 \vee \mathcal{S}_1 \vee \mathcal{S}_2 \vee \ldots)$.

For each iteration of the algorithm, we eliminate one quantifier so we repeat the algorithm for the next quantifier γ_{n-1}, if any, and for the formula $\mathcal{P}_{n-1}$ that is the result of the immediate preceding iteration until the last quantifier γ_1 is eliminated and $\mathcal{P}_0$ is completely in $\mathcal{MHC}$ form.

The formula resulting from each iteration is $\mathcal{P}_n$ where n is the number of the iteration of the algorithms that is equal to the number of quantifiers already eliminated at this point. $\mathcal{P}_{n-1}$, where $n \geq 1$, is considered in $\mathcal{MHC}$ syntax. Thus, $\mathcal{P}_n$ itself is in $\mathcal{MPC}$ syntax but since it is a quantifier-free formula then its atomic formulas are also in $\mathcal{MHC}$ syntax. After the first iteration of the algorithm where $\mathcal{MHC}$ notations will appear it is no longer considered an $\mathcal{MPC}$ formula but rather a $\mathcal{MHC}$ formula. However, $\gamma_n\mathcal{P}_n$, where $n > 0$, at any point of the algorithm where at least one $\mathcal{MPC}$ quantifier exists is in $\mathcal{MHC}^+$ syntax. For instance,

$$\underbrace{\forall x \exists y \forall z \ldots \overbrace{(Q_1 \wedge Q_2 \wedge Q_3 \ldots)}^{\mathcal{MHC}}}_{\mathcal{MHC}^+}$$

All steps of the algorithm are applicable on formulas where $\mathcal{MHC}$ quantifiers do not appear thus we maintain the previously transformed $\mathcal{MHC}$ formulas to be unchanged.

The following examples have different level of complexity to show that using the $\mathcal{MPC} - \mathcal{MHC}$ conversion algorithm we can transform any $\mathcal{MPC}$ formula into $\mathcal{MHC}$ formula.

Example 1:

$$\forall x(xA \rightarrow xB)$$

In this simple formula, x occurs in both the antecedent and the conclusion. We only have to take x as a common factor and then convert $\forall$ to its equivalent form of $\mathcal{MHC}$ quantifier. Thus,

$$\Rightarrow \forall x\, x(A \rightarrow B)$$
$$\Rightarrow [1](A \rightarrow B)$$

Example 2:

$$\forall x(xA \vee yB)$$

In this example, y occurs as a free variable thus $\forall x$ can be pushed inside the clause next to x only,

$$\Rightarrow (\forall x\, xA \vee yB)$$

Since $\forall x\, x \equiv [1]$

$$\therefore [1]A \vee yB$$

yB is an atomic formula that is already equal to its $\mathcal{MHC}$ synonym. Hence, the formula $[1]A \vee yB$ is a $\mathcal{MHC}$ formula.

Example 3:

$$\exists y(xA \wedge yB)$$

The existential quantifier $\exists y$ is corresponding to the bound variable appearing in the second atomic formula. xA is already in its $\mathcal{MHC}$ form thus we push $\exists y$ inside the clause and change it to its equivalent $\mathcal{MHC}$ quantifier,

$$\Rightarrow (xA \wedge \exists y\ yB)$$
$$\Rightarrow xA \wedge \langle 1\rangle B$$

Example 4:

$$\forall x \exists y (xA \rightarrow yB)$$

In this example, both of the antecedent and the conclusion are bounded with different variables. We first convert the implication → into a disjunction,

$$\Rightarrow \forall x \exists y (\neg xA \vee yB)$$

we then push each quantifier next to its corresponding predicate,

$$\Rightarrow (\forall x\ \neg xA \vee \exists y\ yB)$$

In the first part of the formula, we pass the *negation* ¬ through $\forall x$ thus, $\forall x\ \neg xA \equiv \neg \exists x\ xA$,

$$\Rightarrow (\neg \exists x\ xA \vee \exists y\ yB)$$

we then convert the quantifiers to their $\mathcal{MHC}$ forms

$$\Rightarrow (\neg \langle 1\rangle A \vee \langle 1\rangle B)$$

we reverse the disjunction to an implication, thus we reverse the formula into its original form,

$$\therefore \langle 1\rangle A \rightarrow \langle 1\rangle B$$

Notice that the universal quantifier $\forall x$ has changed to an existential quantifier, $\langle 1\rangle$, when the formula is transformed into its $\mathcal{MHC}$ form. If we use the original formula and the outcome formula in a sentence then we will be able to realize how they are logically equal. For Example, $\forall x \exists y (xA \rightarrow yB)$ can be read as

"For all x's and for some y's, if x is an Apple then y is a Blueberry"

while $\langle 1\rangle A \rightarrow \langle 1\rangle B$ can be read as

"If something is an Apple then something is a Blueberry"

Example 5:

$$\forall x(xA \wedge yB \rightarrow xC)$$

To transform this formula, we do not have to convert it to its CNF because when we convert the *implication* $\rightarrow$ into a *disjunction*, the *negation* in the antecedent will be distributed on both the antecedent's atomic formulas and change the conjunction into a disjunction hence we will be able to rearrange the atomic formulas of this formula and push the quantifier $\forall x$ next to its corresponding clause as follows:

$$\Rightarrow \forall x(\neg(xA \wedge yB) \vee xC)$$
$$\Rightarrow \forall x((\neg xA \vee \neg yB) \vee xC)$$
$$\Rightarrow \forall x((\neg xA \vee xC) \vee \neg yB)$$
$$\Rightarrow \forall x(x(\neg A \vee C) \vee \neg yB)$$
$$\Rightarrow (\forall x\, x(\neg A \vee C) \vee \neg yB)$$
$$\Rightarrow [1](\neg A \vee C) \vee \neg yB$$

We then reverse the disjunction into implication, similar to the original formula,

$$\Rightarrow \neg[1](\neg A \vee C) \rightarrow \neg yB$$
$$\Rightarrow \langle 1\rangle\neg(\neg A \vee C) \rightarrow \neg yB$$
$$\Rightarrow \langle 1\rangle(A \wedge \neg C) \rightarrow \neg yB$$

Since every implication is equal to its contrapositive, namely $\alpha \rightarrow \beta \equiv \neg\alpha \rightarrow \neg\beta$, then

$$\Rightarrow \neg(\langle 1\rangle(A \wedge \neg C)) \rightarrow \neg(\neg yB)$$
$$\Rightarrow [1]\neg(A \wedge \neg C) \rightarrow yB$$
$$\Rightarrow [1](\neg A \vee C) \rightarrow yB$$

Example 6:

$$\forall x\exists y(xA \wedge yB \rightarrow xC \vee yD)$$

In this example, both x and y occur once in the antecedent and once in the conclusion. We eliminate $\exists y$ and then $\forall x$. We first convert the *implication*,

$$\Rightarrow \forall x\exists y(\neg(xA \wedge yB) \vee (xC \vee yD))$$

Then we distribute the *negation* in the antecedent,

$$\Rightarrow \forall x\exists y((\neg xA \vee \neg yB) \vee (xC \vee yD))$$

We rearrange the atomic formulas according to their bound variables,

$$\Rightarrow \forall x\exists y((\neg yB \vee yD) \vee (\neg xA \vee xC))$$

$$\Rightarrow \forall x \exists y (y(\neg B \vee D) \vee (\neg xA \vee xC))$$
$$\Rightarrow \forall x (\exists y\ y(\neg B \vee D) \vee (\neg xA \vee xC))$$
$$\Rightarrow \forall x (\langle 1 \rangle (\neg B \vee D) \vee (\neg xA \vee xC))$$
$$\Rightarrow \forall x ((\neg xA \vee xC) \vee \langle 1 \rangle (\neg B \vee D))$$
$$\Rightarrow \forall x (x(\neg A \vee C) \vee \langle 1 \rangle (\neg B \vee D))$$
$$\Rightarrow (\forall x\ x(\neg A \vee C) \vee \langle 1 \rangle (\neg B \vee D))$$
$$\Rightarrow ([1](\neg A \vee C) \vee \langle 1 \rangle (\neg B \vee D))$$
$$\Rightarrow (\neg [1](\neg A \vee C) \rightarrow \langle 1 \rangle (\neg B \vee D))$$
$$\Rightarrow (\langle 1 \rangle \neg (\neg A \vee C) \rightarrow \langle 1 \rangle (\neg B \vee D))$$
$$\therefore (\langle 1 \rangle (A \wedge \neg C) \rightarrow \langle 1 \rangle (\neg B \vee D))$$

Example 7:

$$\forall x (xA \rightarrow \exists y (yB \wedge xC))$$

Before transforming into $\mathcal{MHC}$ form, we first have to convert the formula into its Prenex form since neither x nor y occur as free variables. We convert the *implication* into a *disjunction* and then take the quantifier $\exists y$ outside the bigger clause,

$$\Rightarrow \forall x (\neg xA \vee \exists y (yB \wedge xC))$$
$$\Rightarrow \forall x \exists y (\neg xA \vee (yB \wedge xC))$$

Since we are eliminating an *existential* quantifier and the formula in its DNF then we push the quantifier into the smaller clause where the variable y occurs,

$$\Rightarrow \forall x (\neg xA \vee \exists y (yB \wedge xC))$$
$$\Rightarrow \forall x (\neg xA \vee \exists y ((yB) \wedge xC))$$
$$\Rightarrow \forall x (\neg xA \vee (\exists y\ y(B) \wedge xC))$$
$$\Rightarrow \forall x (\neg xA \vee (\langle 1 \rangle B \wedge xC))$$

Now we eliminate $\forall x$ thus we convert the formula into its CNF and push the $\forall x$ inside both clauses since x occurs in both,

$$\Rightarrow \forall x ((\neg xA \vee \langle 1 \rangle B) \wedge (\neg xA \vee xC))$$
$$\Rightarrow (\forall x (\neg xA \vee \langle 1 \rangle B) \wedge \forall x (\neg xA \vee xC))$$
$$\Rightarrow (\forall x (x(\neg A) \vee \langle 1 \rangle B) \wedge \forall x (x(\neg A \vee C)))$$
$$\Rightarrow ((\forall x\ x(\neg A) \vee \langle 1 \rangle B) \wedge (\forall x\ x(\neg A \vee C)))$$
$$\Rightarrow (([1] \neg A \vee \langle 1 \rangle B) \wedge [1](\neg A \vee C))$$

$$\Rightarrow ((\neg\langle 1\rangle A \vee \langle 1\rangle B) \wedge [1](\neg A \vee C))$$
$$\Rightarrow (\langle 1\rangle A \rightarrow \langle 1\rangle B) \wedge [1](A \rightarrow C)$$

Example 8:

$$\forall x(xA \rightarrow \exists y(yB \wedge \forall z(zC \rightarrow kD)))$$

In this example, we have two inner quantifiers, $\exists y$ and $\forall z$. We will first drag $\forall z$ out and then $\exists y$ as follows:

$$\Rightarrow \forall x(xA \rightarrow \exists y \forall z(yB \wedge (zC \rightarrow kD)))$$
$$\Rightarrow \forall x(\neg xA \vee \exists y \forall z(yB \wedge (zC \rightarrow kD)))$$
$$\Rightarrow \forall x \exists y \forall z(\neg xA \vee (yB \wedge (zC \rightarrow kD)))$$

Since we are eliminating $\forall z$ then we transform the formula into its CNF, and continue for each quantifier,

$$\Rightarrow \forall x \exists y \forall z(\neg xA \vee (yB \wedge (\neg zC \vee kD)))$$
$$\Rightarrow \forall x \exists y \forall z(\neg xA \vee ((yB \wedge \neg zC) \vee (yB \wedge kD)))$$
$$\Rightarrow \forall x \exists y \forall z((\neg xA \vee (yB \wedge \neg zC)) \vee (\neg xA \vee (yB \wedge kD)))$$
$$\Rightarrow \forall x \exists y \forall z(((\neg xA \vee yB) \wedge (\neg xA \vee \neg zC)) \vee ((\neg xA \vee yB) \wedge (\neg xA \wedge kD)))$$
$$\Rightarrow \forall x \exists y \forall z(((\neg xA \vee yB) \wedge (\neg xA \vee \neg zC)) \vee ((\neg xA \vee yB) \wedge (\neg xA \wedge kD)))$$
$$\Rightarrow \forall x \exists y(\forall z((\neg xA \vee yB) \wedge (\neg xA \vee \neg zC)) \vee ((\neg xA \vee yB) \wedge (\neg xA \wedge kD)))$$
$$\Rightarrow \forall x \exists y(((\neg xA \vee yB) \wedge \forall z(\neg xA \vee \neg zC)) \vee ((\neg xA \vee yB) \wedge (\neg xA \wedge kD)))$$
$$\Rightarrow \forall x \exists y(((\neg xA \vee yB) \wedge (\neg xA \vee (\forall z\, z\neg C))) \vee ((\neg xA \vee yB) \wedge (\neg xA \wedge kD)))$$
$$\Rightarrow \forall x \exists y(((\neg xA \vee yB) \wedge (\neg xA \vee ([1]\neg C))) \vee ((\neg xA \vee yB) \wedge (\neg xA \wedge kD)))$$
$$\Rightarrow \forall x((\exists y(\neg xA \vee yB) \wedge (\neg xA \vee ([1]\neg C))) \vee (\exists y(\neg xA \vee yB) \wedge (\neg xA \wedge kD)))$$
$$\Rightarrow \forall x(((\neg xA \vee (\exists y\, yB)) \wedge (\neg xA \vee ([1]\neg C))) \vee ((\neg xA \vee (\exists y\, yB)) \wedge (\neg xA \wedge kD)))$$
$$\Rightarrow \forall x(((\neg xA \vee (\langle 1\rangle B)) \wedge (\neg xA \vee ([1]\neg C))) \vee ((\neg xA \vee (\langle 1\rangle B)) \wedge (\neg xA \wedge kD)))$$
$$\Rightarrow$$
$$((\forall x(\neg xA \vee (\langle 1\rangle B)) \wedge \forall x(\neg xA \vee ([1]\neg C))) \vee (\forall x(\neg xA \vee (\langle 1\rangle B)) \wedge \forall x(\neg xA \wedge kD)))$$
$$\Rightarrow$$
$$(((\forall xx\neg A \vee (\langle 1\rangle B)) \wedge (\forall xx\neg A \vee ([1]\neg C))) \vee ((\forall xx\neg A \vee (\langle 1\rangle B)) \wedge (\forall xx\neg A \wedge kD)))$$
$$\Rightarrow$$
$$((([1]\neg A \vee (\langle 1\rangle B)) \wedge ([1]\neg A \vee ([1]\neg C))) \vee (([1]\neg A \vee (\langle 1\rangle B)) \wedge ([1]\neg A \wedge kD)))$$
$$\Rightarrow$$

$$(((\neg\langle 1\rangle A \vee \langle 1\rangle B) \wedge (\neg\langle 1\rangle A \vee \neg\langle 1\rangle C))) \vee ((\neg\langle 1\rangle A \vee \langle 1\rangle B) \wedge (\neg\langle 1\rangle A \wedge kD)))$$

$$\Rightarrow ((\neg\langle 1\rangle A \vee \langle 1\rangle B) \wedge ((\neg\langle 1\rangle A \vee \neg\langle 1\rangle C) \vee (\neg\langle 1\rangle A \wedge kD))$$

$$\Rightarrow (\neg(\langle 1\rangle A \wedge \neg\langle 1\rangle B) \wedge (\neg(\langle 1\rangle A \wedge \langle 1\rangle C) \vee \neg(\langle 1\rangle A \vee \neg kD))$$

$$\Rightarrow \neg((\langle 1\rangle A \wedge \neg\langle 1\rangle B) \vee ((\langle 1\rangle A \wedge \langle 1\rangle C) \wedge (\langle 1\rangle A \vee \neg kD))$$

5.5. The Monadic Hybrid Calculus $\mathcal{MHC}$ to the Monadic Predicate Calculus $\mathcal{MPC}$ Conversion Algorithm

Algorithm 2: $\mathcal{MHC}$ to $\mathcal{MPC}$ Conversion

Input: A Monadic Hybrid Calculus $\mathcal{MHC}$ formula

Output: A Monadic Predicate Calculus $\mathcal{MPC}$ formula

1. For each atomic formula, $\mathcal{Q}$, in $\mathcal{P}$:
2. If $\mathcal{Q}$ is of the form $x\mathcal{F}$, propagate x through $\mathcal{F}$, as in

$$x(\mathcal{R} \wedge \mathcal{S}) \Rightarrow x\mathcal{R} \wedge x\mathcal{S}$$
$$x(\mathcal{R} \vee \mathcal{S}) \Rightarrow x\mathcal{R} \vee x\mathcal{S}$$
$$x(\mathcal{R} \rightarrow \mathcal{S}) \Rightarrow x\mathcal{R} \rightarrow x\mathcal{S}$$
$$x\neg\mathcal{R} \Rightarrow \neg x\mathcal{R}$$
$$x\mathcal{V} \Rightarrow x\mathcal{V}, \text{ where } \mathcal{V} \text{ is a property variable}$$

3. If an angled brackets, $\langle \quad \rangle$, appears in $\mathcal{Q}$, so that $\mathcal{Q}$ is of the form $\langle\mathcal{R}\rangle\mathcal{S}$, then form a *conjunction* of the two properties, choose a new variable y that has not been previously used in $\mathcal{F}$ to bind it with the Monadic quantifier, $\exists$, and both predicates. Such that if $\mathcal{Q}$ is $\langle\mathcal{R}\rangle\mathcal{S}$ then $\mathcal{Q}'$ will be $\exists y(y\mathcal{R} \wedge y\mathcal{S})$, with y then pushed through $\mathcal{R}$ and $\mathcal{S}$.
4. If a square brackets, $[\quad]$, appears in $\mathcal{Q}$, so that $\mathcal{Q}$ is of the form $[\mathcal{R}]\mathcal{S}$, then form an *implication* where the quantified predicate is the antecedent and the second predicate is the conclusion, choose a new variable z that has not been previously used in $\mathcal{F}$ to bind it with the Monadic quantifier, $\forall$, and both predicates. Such that if $\mathcal{Q}$ is $[\mathcal{R}]\mathcal{S}$ then $\mathcal{Q}'$ will be $\forall z(z\mathcal{R} \rightarrow z\mathcal{S})$, with z then pushed through $\mathcal{R}$ and $\mathcal{S}$.

5.6. $\mathcal{MHC}$ to $\mathcal{MPC}$ Conversion Algorithm Illustration

Transforming a formula from the Monadic Hybrid Calculus to the Monadic Predicate Calculus is a direct process. We transform all atomic formulas of the original formula by applying the corresponding steps in the algorithm.

In step 2 of the algorithm, for any formula of the form $x\mathcal{F}$ where x is a variable we propagate x through $\mathcal{F}$ so that each formula in $\mathcal{F}$ will be bounded with x. Thus, when we transform $\mathcal{F}$ into its $\mathcal{MHC}$ equivalent x will not be disregarded.

In step 3, if a $\mathcal{MHC}$ *existential* brackets, $\langle\ \ \rangle$, exists in the formula where a property variable is quantified as in $\langle\mathcal{R}\rangle\mathcal{S}$, then we conjoin the two properties $\mathcal{R}$ and $\mathcal{S}$ and choose a new variable, y, that has not been already used in the formula to bind it with the equivalent form of $\mathcal{MPC}$ quantifier, $\exists$. Thus, the equivalent formula of $\langle\mathcal{R}\rangle\mathcal{S}$ is $\exists y(y\mathcal{R} \wedge y\mathcal{S})$.

If we use the formula $\langle\mathcal{R}\rangle\mathcal{S}$ in a sentence then it is read, for example, as:

"Some raspberry pies are sugar free"

The sentence clearly indicates that something in the domain of discourse is a *raspberry pie* and *sugar free.* The formula $\exists y(y\mathcal{R} \wedge y\mathcal{S})$ can also be read as:

For some y, y is a raspberry pie and y is sugar free

This is correct if we assume that our syntax includes bound variables otherwise it is not logically valid.

In step 4, if a $\mathcal{MHC}$ *universal* brackets, $[\ \ ]$, exists in the formula where a property variable is quantified, as in $[\mathcal{R}]\mathcal{S}$, then the quantified property will be the antecedent of a Boolean *implication* and the other property will be the consequence. We also choose a variable, z, that has not been already used in the formula and bind it with both properties and the equivalent form of $\mathcal{MPC}$ quantifier, $\forall$. Thus, the equivalent formula of $[\mathcal{R}]\mathcal{S}$ will be $\forall z(z\mathcal{R} \rightarrow z\mathcal{S})$.

Using $[\mathcal{R}]\mathcal{S}$ in the previous example, the sentence will be read as:

"Every raspberry pie is sugar free"

This is equivalent to

"If z is a raspberry pie then z is sugar free"

Since we are transforming atomic formulas from $\mathcal{MHC}$ to $\mathcal{MPC}$ directly then the transformation process will take only one iteration thus the syntax of $\mathcal{MHC}^{+}$ will not be used.

Example:

$$[A]B \wedge \langle C\rangle D \rightarrow x(E \vee F)$$

In this formula, we have $\mathcal{MHC}$ quantifiers thus we will choose two variables other than x that is already in use so our variables are y and z. We transform directly to the $\mathcal{MPC}$ logically equivalent formula that is

$$(\forall y(yA \rightarrow yB) \wedge \exists z(zC \wedge zD)) \rightarrow (xE \vee xF)$$

5.7. $\mathcal{MPC}$ to $\mathcal{MHC}$ Translation Examples

By comparing Monadic Predicate Calculus $\mathcal{MPC}$ formulas with their synonyms formulas in the Monadic Hybrid Calculus $\mathcal{MHC}$ we can clearly identify the differences between the two systems and notice the benefits of using the Monadic Hybrid Calculus system over the Monadic Predicate Calculus. In $\mathcal{MPC}$, we have to clearly describe the domain of discourse that the natural language sentence is applied to. However, in $\mathcal{MHC}$, it is not the case. Either we identify the domain of discourse or not it is already understood from the formalization of the sentence that the domain of discourse is already specified.

The following examples are taken from different textbooks to show that for each formula that is formalized in $\mathcal{MPC}$ there is a logically equivalent formula in $\mathcal{MHC}$. We only extract formulas that are "monadic" (formulas with only one argument) as they are the scope of this dissertation. In each example, we first show the original formula with its original notations and then show its equivalent in $\mathcal{MHC}$ format.

The following examples take the order: sentence, its $\mathcal{MPC}$ formula and then its $\mathcal{MHC}$ formula:

Sentence	$\boldsymbol{\mathcal{MPC}}$	$\boldsymbol{\mathcal{MHC}}$
Everyone is easygoing [5]:		
	$(\forall x)Ex$	$[1]E$

In $\mathcal{MPC}$, the sentence is formalized as $(\forall x)Ex$. First, the variable x was selected to bind with the property E. This is because in $\mathcal{MPC}$ it is necessary to identify our domain of discourse, which is in this case a group of people. In $\mathcal{MHC}$, we do not bind a variable since the domain of discourse will be immediately identified when using the property "easygoing" which clearly refers to a group of people and exclude all other domains. Thus, we write $[1]E$, meaning that any individual in the domain of discourse must possess the property of being an "easygoing" and is included in the formula's context.

Something is out of place [5]:

$$(\exists x)Ox \qquad\qquad \langle 1\rangle O$$

Likewise, in the first formula the variable x is selected to represent a "thing" in the domain of discourse but in the second $\mathcal{MHC}$ formula we assume that anything that fits the property of being "out of place" is meant by the sentence. The formulations are both simple but the $\mathcal{MPC}$ version required a name ("x") for the thing that is out of place.

Either everyone is easy going or no one is:

$$(\forall y)Ey \vee {\sim}(\exists w)Ew \qquad [1]E \vee \neg\langle 1\rangle E$$

In the first formula, two variables were selected, y and w, although it is possible to use the same variable for both formulas. In $\mathcal{MHC}$, we do not use bound variables so we quantify the truth-value 1 to denote every individual that the property "easygoing" applied to. Also, we use $\neg$ for negation instead of ${\sim}$.

If Rita is easygoing, everyone is [5]:

$$Er \supset (\forall w)Ew \qquad\qquad rE \rightarrow [1]E$$

In this conditional sentence, the individual *Rita* is used with the property *easygoing*, so Er is the antecedent, and in the conclusion the variable w was selected as the bound variable. In the $\mathcal{MHC}$ formula, the individual *Rita* precedes the property E, the implication is an arrow $\rightarrow$ instead of the superset $\supset$ and we quantify the truth-value 1 so any individual that can be *easygoing* is subject to this conditional statement context.

Not everyone is easygoing but everyone is ambitious [5]:

$$\sim(\forall z)Ez \,\&\, (\forall y)Ay \qquad\qquad \neg[1]E \wedge [1]A$$

In this conjunction sentence, there are two properties and the variables z and y were selected to denote individuals. In $\mathcal{MHC}$, we quantify the truth-value 1 to refer to any individual in the domain of discourse that can afford being "easygoing" and "ambitious" thus we avoid the usage of bound variables.

If anyone is ambitious, Michael is [5]:

$$(\exists w)Aw \supset Am \qquad\qquad \langle 1\rangle A \rightarrow mA$$

In this sentence, there is a bound variable w in the antecedent and the individual *Michael* is in the conclusion but both of them possess the same property. In $\mathcal{MHC}$ formula, we quantify the truth-value 1 in the antecedent to refer to *anyone* in the domain of discourse and we add the individual *Michael* to the conclusion where both the antecedent and the conclusion possess the property of being *Ambitious.*

All dolphins are mammals [5]:

$$(\forall z)(Dz \supset Mz) \qquad\qquad [D]M$$

In this sentence, we can clearly see the benefits of using the Monadic Hybrid Calculus over the Monadic Predicate Calculus. In the first formula, the variable z was bounded with the property of being *Dolphin* and another part was added to the formula to bind the variable z with the property if being *Mammal.* In $\mathcal{MHC}$, we already know from the sentence's context that the species *Dolphins* is the range of the quantifier. So we quantify the property of being a *Dolphin* and attach it to the property of being a *Mammal* meaning that anything in the world that is a *Dolphin* is also a *Mammal.*

All reptiles are cold-blooded [5]:

$$(\forall w)(Rw \supset Cw) \qquad\qquad [R]C$$

Similar to the previous example, the variable w was used twice with both properties, *reptiles* and *cold-blooded,* in the antecedent and in the conclusion. In the $\mathcal{MHC}$ formula, we quantify the property of being *reptile* species first meaning that being a *reptile* is also being a *cold-blooded.*

Grizzly bears are dangerous and polar bears are dangerous, but black bears are not. [5]:

$$[(\forall w)(Gw \supset Dw)\ \&\ (\forall w)(Pw \supset Dw)]\ \&\ (\forall w)(Bw \supset {\sim}Dw)$$

$$[G]D \wedge [P]D \wedge [B]\neg D$$

In this example, the variable w has to be repeatedly used with all properties of the sentence, *Grizzly*, *Polar*, *Black* and *Dangerous.* This makes the symbolization of this sentence more complicated and extended. In $\mathcal{MHC}$, we assume that any bear that is *Grizzly*, *Polar* or *Black* fills into the categories of being either *Dangerous* or *Not dangerous.* In other words, our domain of discourse is already known to include *bears* that are *Grizzly*, *Polar* or *Black.* Thus, we assume that the property of being *Dangerous* logically fill into this group of individuals. As a result, we obtain a much shorter and simpler symbolization of the original sentence.

Every integer is either odd or even [5]:

$$(\forall y)(Oy \vee Ey) \qquad\qquad [I](O \vee E)$$

In the $\mathcal{MPC}$ formula, the universe of discourse was identified (in the reference textbook) to be the set of *integer numbers* otherwise the property of being an *integer* number must be used in the formalization of this sentence which will add to the complexity of the formula. In $\mathcal{MHC}$, we already quantify the group of *integer numbers, I,* thus we do not have to separately introduce our domain of discourse in a prefix dictionary.

There exists a natural number divisible by 2 and y is divisible by 3 [6]:

$$(\exists x)(A(x).B(\mathrm{y})) \qquad\qquad \langle N\rangle A \wedge yB$$

In the $\mathcal{MPC}$ formula of this example, A is the property of being *divisible by 2* and B is the property of being *divisible by 3, x* is the bound variable and y is just an individual. In the $\mathcal{MHC}$ formula, since the second sentence specifically named y as a constant then we cannot avoid using it since it is not a *bound* variable like x.

Someone is happy. The happy are lucky. Therefore someone is lucky [7]:

$$(\exists x)Hx, (\forall x)(Hx \rightarrow Lx) \vdash (\exists x)Lx \qquad \langle 1\rangle H, [H]L, \therefore \langle 1\rangle L$$

In this example, when using $\mathcal{MHC}$, we quantify the truth-value 1 twice in both of the sentences where the phrase *someone* appears thus following $\mathcal{MHC}$ syntax the formula is much more concise.

Somebody is happy. So it is false that everybody is unhappy [7]:

$$(\exists x)Hx \vdash \neg(\forall x)\neg Hx \qquad \langle 1\rangle H, \therefore \neg[1]\neg H$$

In this similar example, the truth-value *1* is quantified in the $\mathcal{MHC}$ formula as it is already known that the domain of discourse is people who are alive.

There is at least one thing in the universe such that it is a painter, and if it is a craftsman and it is a decorator then it has an eye for color balance; and it is not an artist. [8]:

$$(\exists x)(\{Px \cdot [(Cx \cdot Dx) \supset Bx]\} \cdot {\sim}Ax) \qquad \langle P\rangle((C \wedge D) \rightarrow B) \wedge \neg A)$$

This is a complicated sentence to formulize. In $\mathcal{MPC}$, the variable x is repeatedly used with every property. Also, different parentheses, [ ],( ) and { }, are used to correctly combine segments of antecedents and consequences. In $\mathcal{MHC}$, we only use the parentheses () to combine properties as well as quantify the property P. As a result, the formula is clearer and less complex.

Some persons are rich and kind and some persons are neither rich nor kind [8]:

$$(\exists x)[Px \cdot (Rx \cdot Kx)] \cdot (\exists x)[Px \cdot {\sim}(Rx \vee Kx)]$$

$$\langle P\rangle(R \wedge K) \wedge \langle P\rangle(\neg R \wedge \neg K)$$

In this example, the property of being a *person* is specified thus when we use $\mathcal{MHC}$ to formulize the sentence we have to use the property P in the antecedent. We also used "$\neg R \wedge \neg K$" instead of "${\sim}(Rx \vee Kx)$" but both hold the same logical content.

No students who are lazy will pass and no students who will pass are lazy [8]:

$$(x)[(Sx \cdot Lx) \supset {\sim}Px] \cdot (x)[(Sx \cdot Px) \supset {\sim}Lx]$$

$$[S \wedge L]\neg P \wedge [S \wedge P]\neg L$$

These two sentences state the opposite of each other in terms of using properties. In $\mathcal{MHC}$, it is possible to combine two or more properties inside one quantifier such as in $[S \wedge L]$. Thus, $[S \wedge L]\neg P$ means *all who are student and lazy will not pass.*

Poets and painters are conservative neither in their dress nor in their politics [9]:

$$(x)((F(x) \vee G(x)) \supset {\sim}\big(H(x) \vee I(x)\big)) \qquad\qquad [F \vee G]\neg(H \vee I)$$

In this example, when we used the $\mathcal{MHC}$ system we could quantify the properties *F* and *G* together as well as we use the parentheses to combine the two properties in the consequent.

Some people are neither honest nor truthful [10]:

$$(\exists x)[Px \cdot {\sim}(Hx \vee Tx)] \qquad\qquad \langle P\rangle(\neg H \wedge \neg T)$$

In the $\mathcal{MHC}$ formula, we treat the property of being *one of the people* different than the $\mathcal{MPC}$ formula. The property P is quantified in $\mathcal{MHC}$ thus it is meant for any individual from the society that is *neither honest nor truthful*.

Boston is industrial and Boston is a city [11]:

$$Ib \cdot Cb \qquad\qquad b(I \wedge C)$$

In the $\mathcal{MPC}$ formula, the individual *Boston* has been duplicated with both of the properties *industrial* and *city*. In the $\mathcal{MHC}$ formula, the individual *Boston* is a common factor for both of the properties.

Each thing in the universe is such that if it is a drug and it is habit-forming, then if it is not prescribed by a physician then it is not the case that either it is safe or it is beneficial [11]:

$$(x)\{(Dx \cdot Fx) \supset [\sim Px \supset \sim(Sx \vee Bx)]\} \qquad [D \wedge F \wedge \neg P]\neg(S \vee B)$$

In case of multiple implications sentence, the $\mathcal{MHC}$ formula combines multiple properties inside one quantifier in addition to the negation symbol. Thus, $[D \wedge F \wedge \neg P]$ means *all habit-forming drugs not prescribed by a physician.*

There is at least one thing in the universe such that it is a student and it is not the case that either it is clever or it is industrious, and it is lazy and it is dense, and either it will pass or it will graduate. [11]:

$$(\exists x)(\{Sx \cdot [\sim(Cx \vee Ix) \cdot (Lx \cdot Dx)]\} \cdot (Px \vee Gx))$$

$$\langle S \wedge L \wedge D\rangle\big(\neg(C \vee I) \wedge (P \vee G)\big)$$

In this long sentence, the formalization without binding a variable in $\mathcal{MHC}$ makes the formula more concise and less complex. Also, when using only the parentheses () we manage to combine different antecedents and consequences without affecting the overall meaning of the sentence.

In the following example, the formalization of a group of sentences is different and that is according to the two given domains of discourse. The first domain is "A) Marbles", and the second one is "B) Marbles and marble players". Since we have two groups of individuals, *marbles* and *players*, the formalization of the sentences will be as follows:

All the marbles are blue or all the marbles are green [5]:

In domain A, two variables, z and y, were selected although one variable is enough:

$$(\forall z)Bz \vee (\forall y)Gy \qquad\qquad [1]B \vee [1]G$$

In domain B, only one variable was selected thus:

$$(\forall y)(My \supset By) \vee (\forall y)(My \supset Gy) \qquad [M]B \vee [M]G$$

Some of the marbles are blue and some are green, but none is red [5]:

We have in domain A:

$$[(\exists x)Bx \,\&\, (\exists x)Gx] \,\&\, {\sim}(\exists x)Rx$$

and in domain B:

$$[(\exists x)(Mx \,\&\, Bx) \,\&\, (\exists x)(Mx \,\&\, Gx)] \,\&\, {\sim}(\exists x)(Mx \,\&\, Rx)$$

In the $\mathcal{MPC}$ we ended up with two formulas that change according to the domain of discourse. In $\mathcal{MHC}$, since we do not deal with bound variables we will have only one formula, which is:

$$\langle M\rangle B \wedge \langle M\rangle G \wedge \neg\langle M\rangle R$$

Similarly to the previous example:

If any marbles is blue they all are [5]:

In domain A:

$$(\exists w)Bw \supset (\forall x)Bx$$

And in domain B:

$$(\exists w)(Mw \,\&\, Bw) \supset (\forall x)(Mx \supset Bx)$$

In $\mathcal{MHC}$:

$$\langle M\rangle B \rightarrow [M]B$$

We can clearly see the confusion caused by using bound variables in every domain but in $\mathcal{MHC}$ since we do not use bound variables we skipped this confusion thus we only have one formula.

The Monadic Hybrid Calculus works perfectly with syllogisms. The formalization process is easier, less complicated and makes the syllogisms shorter as well as variables-free.

Example1 [12]:

Each α is a β	$(\forall x)[\alpha(x) \Rightarrow \beta(\mathrm{x})]$	$[A]B$
Some α is a β	$(\exists x)[\alpha(x) \wedge \beta(\mathrm{x})]$	$\langle A\rangle B$
No α is a β	$(\forall x)[\alpha(x) \Rightarrow \tilde{\beta}(\mathrm{x})]$	$[A]\neg B$
Some α is not a β	$(\exists x)[\alpha(x) \wedge \tilde{\beta}(\mathrm{x})]$	$\langle A\rangle\neg B$

In this example, because we do not use variables, we only use uppercase letters for properties thus instead of using lowercase alpha α we use uppercase alpha A and instead of using lowercase beta β we use uppercase beta B.

Example2 [13]:

All integers are real numbers.	$(x)[I(x) \rightarrow R(\mathrm{x})]$	$[I]R$
All real numbers are complex numbers	$(x)[R(x) \rightarrow C(\mathrm{x})]$	$[R]C$
$\therefore$ All integers are complex numbers	$\therefore (x)[I(x) \rightarrow C(\mathrm{x})]$	$\therefore [I]C$

In this example, the $\mathcal{MHC}$ system works very well and it represents the sentences clearly and in a concise description of premises and conclusions therefore it is easier to verify the validity of the syllogisms.

5.8. A Compactness Proof for the Monadic Hybrid Calculus $\mathcal{MHC}$

Since we proved that the $\mathcal{MHC}$ is equivalent to the $\mathcal{MPC}$ then every model in $\mathcal{MPC}$ has a model in $\mathcal{MHC}$ accordingly we prove the Compactness of $\mathcal{MHC}$.

We shall use $\vDash$ as the satisfaction symbol, so that $\mathcal{I} \vDash \delta$ means, for instance, the interpretation $\mathcal{I}$ satisfies the set of formulas δ by satisfying every element of δ, and use $\vdash$

as a derivation symbol, so that $\delta \vdash \psi$ means, for instance, the conclusion ψ can be derived from the set of axioms δ using the tableau.

Definition 5.8.1 *(Compactness)*: For any set Δ of formulas, Δ is consistent if and only if every finite subset of Δ is consistent.

Proof 5.8.2. Let Δ be a set of Monadic Hybrid Calculus $\mathcal{MHC}$ formulas all of whose finite subsets are satisfiable. For every $\delta \in \Delta$, there is an equivalent Monadic Predicate Calculus formula δ'. Let $\Delta' = \{\delta' | \delta \in \Delta\}$. Clearly, Δ' is finitely satisfiable. By the Monadic Predicate Calculus compactness, if an interpretation $\mathcal{I}$ satisfy Δ' then, by equivalence, $\mathcal{I}$ will also satisfies Δ.

Chapter 6: The Hybrid Predicate Calculus $\mathcal{HPC}$

In this chapter, we formally introduce the General Hybrid Predicate Calculus $\mathcal{HPC}$ system [14] by presenting its formal syntax and semantics. The general Hybrid Predicate Calculus is the complete system of the Hybrid Predicate Calculus $\mathcal{HPC}$ where formulas use an unlimited number of arguments. We explain the arity of the $\mathcal{HPC}$'s predicates and relations as well as its new notions, the *tilde*, the *slash* and the *asterisk*. We introduce each notion's formal semantics along with its pragmatic usage.

6.1. The Formal Syntax of the Hybrid Predicate Calculus $\mathcal{HPC}$

The alphabet of the Hybrid Predicate Calculus $\mathcal{HPC}$ consists of:

I. An infinite group of upper case letters *{A, B, C, ...}* that represents predicates. Each predicate is assumed to possess infinite arity.

II. An infinite group of lower case letters *{x, y, z, ...}* that are individual constants.

III. Logical symbols: $\wedge$ for *conjunction*, $\vee$ for *disjunction*, $\rightarrow$ for *implication*, $\neg$ for *negation*, square brackets around predicate, $[P]$, for *universal quantifier*, angle brackets around predicate, $\langle P\rangle$, for *existential quantifier*, ~ for reversed predicate, / for reflexive predicate, * for numeral shifting (increment) of predicate's subjects, the apostrophe ($'$) to control the effect of the ~, / and * on individuals of the formula and ( ) parentheses to assemble two or more predicates.

Definition 6.1.1. A Hybrid Predicate Calculus formula is either:

i. 0 or 1, a predicate constant,

ii. $\mathcal{v}\mathcal{F}$, where $\mathcal{F}$ is a property formula and $\mathcal{v}$ is a constant. Such as: xA *"x is a Student",*

iii. $[\mathcal{A}]\mathcal{P}$, where $\mathcal{A}$ and $\mathcal{P}$ are both formulas with *universal* quantification for the first property. Such as: $[A](R \vee G)$ *"Anything that is an Apple is either Red or Green"*,

iv. $\langle\mathcal{A}\rangle\mathcal{P}$, where $\mathcal{A}$ and $\mathcal{P}$ are both property formulas with *existential* quantification

for the first property. Such as: $\langle A\rangle(R \wedge G)$ *"Some things that are Apple are Red or Green"*,

v. A Boolean combination using logical symbols, namely $\wedge$ for *conjunction*, $\vee$ for *disjunction* or $\rightarrow$ for *implication*, to connect two or more formulas. Such that if $\mathcal{G}$ and $\mathcal{H}$ are formulas so are $(\mathcal{G} \wedge \mathcal{H}), (\mathcal{G} \vee \mathcal{H}), (\mathcal{G} \rightarrow \mathcal{H})$,

vi. Boolean negation $\neg$, so that if $\mathcal{R}$ is a formula so is $\neg\mathcal{R}$. For instance, $\neg[A]G$, *"Not all Apples are Green",*

vii. Truth constants: 0 for *False* and 1 for *True.*

Note: Unlike the $\mathcal{MHC}$, there are no property formulas in Hybrid Predicate Calculus, only absolute formulas.

6.2. The Formal Semantics of the Hybrid Predicate Calculus $\mathcal{HPC}$

Definition 6.2.1. An interpretation $\mathcal{I}$ is a function that assigns to 1 a nonempty domain of discourse $\mathcal{I}(1)$-a set of individuals (people, things, days, …etc.), that assigns to each predicate constant $\mathcal{V}$ a set $\mathcal{I}(\mathcal{V})$ of infinite sequence of individuals and that assigns to each individual variable $\mathcal{v}$ an element $\mathcal{I}(\mathcal{v})$ of $\mathcal{I}(1)$.

The formal semantics of the Hybrid Predicate Calculus $\mathcal{HPC}$ is as follows:

$\mathcal{I}, a_0, a_1, a_2, \ldots \vDash \mathcal{V}$	*iff*	$\langle a_0, a_1, a_2, \ldots\rangle \in \mathcal{I}(\mathcal{V})$
$\mathcal{I}, a_0, a_1, a_2, \ldots \vDash x\mathcal{P}$	*iff*	$\mathcal{I}, \mathcal{I}(x), a_0, a_1, a_2, \ldots \vDash \mathcal{P}$
$\mathcal{I}, a_0, a_1, a_2, \ldots \vDash [\mathcal{A}]\mathcal{P}$	*iff*	for all i, if $\mathcal{I}, i, a_0, a_1, a_2, \ldots \vDash \mathcal{A}$ then $\mathcal{I}, i, a_0, a_1, a_2, \ldots \vDash \mathcal{P}$
$\mathcal{I}, a_0, a_1, a_2, \ldots \vDash \langle\mathcal{A}\rangle\mathcal{P}$	*iff*	for some i, if $\mathcal{I}, i, a_0, a_1, a_2, \ldots \vDash \mathcal{A}$ then $\mathcal{I}, i, a_0, a_1, a_2, \ldots \vDash \mathcal{P}$
$\mathcal{I}, a_0, a_1, a_2, \ldots \vDash (\mathcal{P} \vee \mathcal{Q})$	*iff*	$\mathcal{I}, a_0, a_1, a_2, \ldots \vDash \mathcal{P}$ or $\mathcal{I}, a_0, a_1, a_2, \ldots \vDash \mathcal{Q}$
$\mathcal{I}, a_0, a_1, a_2, \ldots \vDash (\mathcal{P} \wedge \mathcal{Q})$	*iff*	$\mathcal{I}, a_0, a_1, a_2, \ldots \vDash \mathcal{P}$ and $\mathcal{I}, a_0, a_1, a_2, \ldots \vDash \mathcal{Q}$
$\mathcal{I}, a_0, a_1, a_2, \ldots \vDash \neg\mathcal{P}$	*iff*	$\mathcal{I}, a_0, a_1, a_2, \ldots \nvDash \mathcal{P}$
$\mathcal{I}, a_0, a_1, a_2, \ldots \vDash (\mathcal{P} \rightarrow \mathcal{Q})$	*iff*	if $\mathcal{I}, a_0, a_1, a_2, \ldots \vDash \mathcal{P}$ then $\mathcal{I}, a_0, a_1, a_2, \ldots \vDash \mathcal{Q}$

6.3. The Arity of the Hybrid Predicate Calculus $\mathcal{HPC}$ Predicates

Predicates in the $\mathcal{HPC}$ system are treated differently than other formal logic systems. One of the important ideas in the Hybrid Predicate Calculus is to treat every *predicate* as if it possesses an infinite ω-sequence of arguments. The arguments are represented as circled numbers …… ,② ,①,⓪ that are in a descending order. Depending on the predicate, we can assume any number of arguments that this predicate can take. For example, the property of being a *Car* is a one argument predicate thus if C represents the property *Car* then C is treated as

⓪ *is a Car*

where the circled number ⓪ can be replaced by any individual.

Another example, if we have the predicate *registered with* then this predicate will take two arguments thus if R denotes the predicate *registered with* then R is treated as

① *is registered with* ⓪

where the two circled numbers ① and ⓪ represent the individuals that the predicate R can take.

If we have a predicate such as *bought…for* that takes three arguments and B denotes this predicate then B is treated as

② *bought* ① *for* ⓪

In the first example, ⓪ *is a Car,* if we replace ⓪ with any individual, say x, then xC is read as:

x *is a Car*

In the second example, since the predicate R can accommodate two arguments then R could be as in one of the following example forms:

bR	⓪ *is registered with Bill*
obR	*Omar is registered with Bill*
obR	*Omar is taking a course from Bill*
$[S]bR$	*Every student is registered with Bill*
$\langle S\rangle bR$	*Bill has some students registered in his course*

$\langle S\rangle\langle P\rangle R$	*Some students are registered with some professors*

In the third example, since the predicate B can accommodate three arguments then B can be as in one of the following example forms:

jB	① bought ② for *John*
sjB	② bought a *sedan* for *John*
$[A]jB$	② bought all the *apples* for *John*
$\langle S\rangle c\langle L\rangle B$	*Some student* bought a *computer* for *some laboratories*
$\langle A\rangle\langle S\rangle B$	② bought *some apple* for *some student*

Notice that in all of the examples the circled numbers are written in a descending order no matter where the individuals are located in the sentence.

6.4. The Tilde "~" Operator

The *tilde,* ~, operator is one of the novel operators that has been introduced in the Hybrid Predicate Calculus $\mathcal{HPC}$. The operator ~ denotes the passive voice in natural language sentences. For example, the active voice sentence

x likes y

is denoted as

xyL

where *L* denotes the relation *likes*. If we convert the sentence to its a passive voice

y is liked by x

then we formalize it using ~ as

$yx{\sim}L$

the $\sim L$ denotes the passive voice relation *is liked by*.

This is not to be confused with *tilde* (~) used by some logicians to denote *negation*. In the Hybrid Predicate Calculus, however, we use ¬ to denote *negation*.

The *tilde* operator can convert any relation between free or quantified individuals, however, it is not guaranteed that the same meaning of the original sentence is preserved. For example, all but one of the following implications are correct:

$$xyL \rightarrow yx{\sim}L$$
$$x\langle Y\rangle L \rightarrow \langle Y\rangle x{\sim}L$$
$$x[Y]L \rightarrow [Y]x{\sim}L$$
$$\langle X\rangle yL \rightarrow y\langle X\rangle{\sim}L$$
$$\langle X\rangle\langle Y\rangle L \rightarrow \langle Y\rangle\langle X\rangle{\sim}L$$
$$\langle X\rangle[Y]L \rightarrow [Y]\langle X\rangle{\sim}L$$
$$[X]yL \rightarrow y[X]{\sim}L$$
$$[X]\langle Y\rangle L \nrightarrow \langle Y\rangle[X]{\sim}L$$
$$[X][Y]L \rightarrow [Y][X]{\sim}L$$

The antecedents in these formulas always imply to their equivalent conclusions where $\sim$ appears except in the case $[X]\langle Y\rangle L \nrightarrow \langle Y\rangle[X]{\sim}L$ where the sentence

Everyone in X likes someone in Y

does not imply

Someone in Y is liked by everyone in X

In what follows, we examine the *logical equivalence* of the previously listed formulas. Thus,

$$xyL \equiv yx{\sim}L$$

The first formula can be read as,

x likes y

The second formula is read as,

y is liked by x

The two formulas hold the same meaning.

Similarly,

$$x\langle Y\rangle L \equiv \langle Y\rangle x{\sim}L$$

"x likes someone in Y" is logically equivalent to *"Someone in Y is liked by x"*

$$x[Y]L \equiv [Y]x{\sim}L$$

"*x likes everyone in Y"* is logically equivalent to *"Everyone in Y is liked by x"*

$$\langle X\rangle yL \equiv y\langle X\rangle{\sim}L$$

"Someone in X likes y" is logically equivalent to *"y is liked by someone in X"*

$$\langle X\rangle\langle Y\rangle L \equiv \langle Y\rangle\langle X\rangle{\sim}L$$

"Someone in X likes someone in Y" is logically equivalent to *"Someone in Y is liked by someone in X"*

$$[X]yL \equiv y[X]{\sim}L$$

"Everyone in X likes y" is logically equivalent to *"y is liked by everyone in X"*

$$[X][Y]L \equiv [Y][X]{\sim}L$$

"Everyone in X likes everyone in Y" is logically equivalent to *"Everyone in Y is liked by everyone in X"*

The only exceptions are:

$$\langle X\rangle[Y]L \not\equiv [Y]\langle X\rangle{\sim}L$$

"Someone in X likes everyone in Y" is not logically equal to *"Everyone in Y is liked by someone in X"*

and

$$[X]\langle Y\rangle L \not\equiv \langle Y\rangle[X]{\sim}L$$

"Everyone in X likes someone in Y" is not logically equal to *"Someone in Y is liked by everyone in X"*

The previously mentioned rules will not be affected if *negation* appears in any place within the formulas. For example:

$$"x\langle Y\rangle{\neg}L \equiv \langle Y\rangle x{\sim}{\neg}L"$$

"x does not like someone in Y" is logically equivalent to *"Someone in Y is not liked by x"*

Also, the rules are not affected by Boolean connectives appearing among two of these formulas, for example:

$$xyL \wedge xzL \equiv yx{\sim}L \wedge zx{\sim}L$$
$$xyL \wedge xzL \equiv yx{\sim}L \wedge zx{\sim}L$$

and

$$[X]yL \wedge [X]\langle Z\rangle L \not\equiv y[X]{\sim}L \wedge \langle Z\rangle[X]{\sim}L$$

So far, we have only seen the *tilde* operator works with two elements. If we have more than two individuals, or quantified individuals, with property, predicate or relation the effect of *tilde* is only imposed on the first two arguments immediately preceding property, predicate or relation. Thus, if a formula has three arguments, for example, $[X]y\langle Z\rangle L$, then with the *tilde* operator the formulas will imply $[X]\langle Z\rangle y{\sim}L$ where $\sim$ has only reversed the last two arguments, y and $\langle Z\rangle$.

To control the effect of *tilde* over other arguments we use the apostrophe ($'$) to indicate the position of arguments being affected. In the formula $[X]y\langle Z\rangle L$, if we want to reverse $[X]$ and y then the formula will be $y[X]\langle Z\rangle{\sim}'L$ where the appearance of only one apostrophe ($'$) indicates that we skipped the argument immediately preceded the relation L and $\sim$ effect is applied on the two other arguments. Accordingly to this concept, if we have a relation with four arguments, for example, $x\langle Y\rangle z\langle K\rangle L$, then using two apostrophes with $\sim$ the passive voice formula will be $\langle Y\rangle xz\langle K\rangle{\sim}''L$ where the effect of $\sim$ is applied on the last two arguments $\langle Y\rangle$ and x.

The *implications* and *logical equivalent* of the list of formulas introduced earlier are all true in the case of more than two arguments occur with the relation. Thus, the following examples are also true:

$$xyzL \rightarrow yxz{\sim}'L$$
$$xyzL \equiv yxz{\sim}'L$$

$$xyzkL \equiv yxzk{\sim}''L$$

$$x\langle Y\rangle zL \rightarrow \langle Y\rangle xz{\sim}'L$$
$$x\langle Y\rangle zL \equiv \langle Y\rangle xz{\sim}'L$$

$$x\langle Y\rangle zkL \equiv \langle Y\rangle xzk{\sim}''L$$

$$x[Y]zL \equiv [Y]xz{\sim}'L$$
$$x[Y]zkL \equiv [Y]xzk{\sim}''L$$

$$\langle X\rangle\langle Y\rangle zL \equiv \langle Y\rangle\langle X\rangle z{\sim}'L$$

Also, the following fails:

$$\langle X\rangle[Y]zL \equiv [Y]\langle X\rangle z{\sim}'L$$

The effect of the *tilde* operator is also applicable on the arity of predicates (the circled numbers). For example, if L denotes the relation *Like* that takes two arguments then L can be read as,

① likes ⓪

and its passive voice $\sim L$ can be read as,

① is liked by ⓪

6.5. The Formal Semantics of the Tilde "~" Operator

Assuming that an interpretation $\mathcal{I}$ is a nonempty domain of discourse represents set of individuals (people, things, days, …etc.) that correspond such subjects, in the domain of discourse, to property variables $\mathcal{V}$, such that $\mathcal{I}(\mathcal{V})$, and individuals to individual variables x, such that $\mathcal{I}(x)$.

The formal semantics for formulas with *tilde* is:

$\mathcal{I}, a_0, a_1, a_2, \ldots \vDash {\sim}\mathcal{P}$ *iff* $\mathcal{I}, a_1, a_0, a_2, \ldots \vDash \mathcal{P}$

6.6. The Forward Slash "/" Operator

The *slash* operator is another novel notation introduced in the Hybrid Predicate Calculus $\mathcal{HPC}$. The main function of *slash* is to eliminate repetition of individuals and represents reflexive verbs in the natural language sentences. For example,

x likes x

which can be rephrased as *x likes him/herself*, is formalized as,

$$xxL$$

where *L* denotes the relation *likes*. Using the *slash* operator, the formula is expressed as,

$$x/L$$

so that $x/\equiv xx$. This concept allows for more flexible sentences and more powerful expressiveness even for predicates that take more than two individual such as in the following examples,

$xyyK$ *x knows that y likes* y

that can be expressed as,

xy/K *x knows that y likes him/herself*

and

K *② knows that ① like ⓪*

that can be expressed as,

$/K$ *① knows that ⓪ like him/herself*

Using the *slash* operator with quantified individuals we will have the following:

$$\langle A\rangle/L \rightarrow \langle A\rangle\langle A\rangle L$$

$\langle A\rangle/L$ is read as,

Someone in A likes him/herself

this is implies,

$$\langle A\rangle\langle A\rangle L$$

that is read as,

Someone in A likes someone in A

which implies that the same individual in *A* likes someone in *A* who is him/herself. However, $\langle A\rangle/L$ is not logically equal to $\langle A\rangle\langle A\rangle L$ because it may be interpreted that the individual in *A* may like a different individual in *A* but not him/herself.

With universal quantifiers the *implication* and *equivalency* are both not true, thus:

$$[A]/L \nrightarrow [A][A]L$$

"Everyone in A likes him/herself" does not imply that *"Everyone in A likes everyone in A"*

Also,

$$[A]/L \not\equiv [A][A]L$$

"Everyone in A likes him/herself" is not logically equal to *"Everyone in A likes everyone in A"*

However, $[A]/L \rightarrow [A]\langle A\rangle L$ is true in case of implication because the first formula is read as,

"Everyone in A likes him/herself"

which implies,

"Everyone in A like someone in A"

thus every individual in *A* likes someone who is him/herself.

Similar to the *tilde* operator, the *slash* operator's effect is applied only on the two arguments immediately preceding the relation. For example,

$$xyyL$$

that is read as

x likes y who likes y

can be re-formalized as

$$xy/L$$

which is read as

x likes y who likes him/herself.

We use the apostrophe ($'$) to indicate which individual is affected by the use of $/$. Thus, using the apostrophe ($'$) with $/$ on the formula,

$$xyyzL$$

will change it into,

$$xyz/'L$$

where the apostrophe indicates that the second individual from the position of the relation L, that is y, is the reflexive. Similarly,

$$xxyzL$$

can be rewritten as

$$xyz/''L$$

Also, in case of quantified individuals,

$$[A][A][B]L \to [A][B]/'L$$

and so forth.

Using apostrophes will not contradict our previously discussed cases of the *slash implications* and *logical equivalency*. Thus, the following are true:

$$\langle A\rangle\langle B\rangle/'L \to \langle A\rangle\langle A\rangle\langle B\rangle L \quad \text{but} \quad \langle A\rangle\langle A\rangle\langle B\rangle L \nrightarrow \langle A\rangle\langle B\rangle/'L$$

$$\langle A\rangle\langle B\rangle\langle C\rangle/''L \to \langle A\rangle\langle A\rangle\langle B\rangle\langle C\rangle L \quad \text{but} \quad \langle A\rangle\langle A\rangle\langle B\rangle\langle C\rangle L \nrightarrow \langle A\rangle\langle B\rangle\langle C\rangle/''L$$

$$[A][B]/'L \to [A][A][B]L \quad \text{but} \quad [A][A][B]L \nrightarrow [A][B]/'L$$

$$[A][B][C]/''L \to [A][A][B][C]L \quad \text{but} \quad [A][A][B][C]L \nrightarrow [A][B][C]/''L$$

6.7. The Formal Semantics of the Forward Slash "/" Operator

The semantics of formulas with the *slash* operator is as follows:

$\mathcal{I}, a_0, a_1, a_2, \ldots \vDash /\mathcal{P}$ *iff* $\mathcal{I}, a_0, a_0, a_1, \ldots \vDash \mathcal{P}$

6.8. The Asterisk "*" Operator

The *asterisk* operator is another new notation introduced by the Hybrid Predicate Calculus $\mathcal{HPC}$. However, its function is not important as the other new notations. The main function of the *asterisk* operator is to shift (increase the count) the number of arguments one step per *asterisk* so elements can be inserted within the formula without disturbing the original syntax. For example, the formula L is logically equivalent to another formula with an *asterisk* operator that is $a * L$. Also, the *asterisk* is used to control the number of arguments in formulas for specific purposes.

To clarify, suppose we have the predicate P with the following dictionary:

P *"② registered with ① who teaches ⓪"*

If we add the individual x to P then

xP *"① registered with ⓪ who teaches x"*

If we add the *asterisk* to P then

$*P$ *"③ registered with ② who teaches ①"*

The *asterisk* increases the arguments number count of the Hybrid Predicate Calculus predicate. Thus, if we add the individual x along with the *asterisk* to P then we end up with the original formula P without disturbing the original count of the arguments inside P. So,

$a*P$ *"② registered with ① who teach ⓪"*

is the same as

P *"② registered with ① who teach ⓪"*

This function of the *asterisk* is beneficial if we want to include, or exclude, individuals inside the formula without disturbing its syntactic order.

The *asterisk* operator can include, or exclude, only one individual. For example, $ab*L$ is equal to aL. The *asterisk* has excluded the b from the original formula.
Using the *asterisk* with a quantifier, it excludes the entire quantified individual. For example,

P *"② registered with ① who teach ⓪"*

is logically equivalent to

$[A]*P$ *"② registered with ① who teach ⓪"*

The *asterisk* operator can only manipulate the individual immediately preceding the predicate. To gain control over all individuals occur in the formula we use the apostrophe ($'$) to refer to the location of the individual being included or excluded. For example,

$$ab*'L \rightarrow bL$$

$$abc * ''L \rightarrow bcL$$

6.9. The Formal Semantics of the Asterisk "*" Operator

The semantics of formula with the *asterisk* operator is as follows:

$\mathcal{I}, a_0, a_1, a_2, \ldots \vDash * \mathcal{P}$ *iff* $\mathcal{I}, a_1, a_2, a_3, \ldots \vDash \mathcal{P}$

6.10. Conjectures

We conjecture that there are Tableau Rules with respect to which the Hybrid Predicate Calculus $\mathcal{HPC}$ is sound and complete but we still do not have enough Tableau Rules to construct these proofs. We also conjecture that the Hybrid Predicate Calculus $\mathcal{HPC}$ can be proven to be equivalent to the Predicate Logic but the scope of this dissertation does not allow us to proceed to this point.

Chapter 7: Summary and Future Work

In this concluding chapter, we summarize our results and describe future work that could be pursued to examine both the Monadic Hybrid Predicate Calculus $\mathcal{MHC}$ and the Hybrid Predicate Calculus $\mathcal{HPC}$.

7.1. Summary

The Monadic Hybrid Calculus $\mathcal{MHC}$ is a new formal system that does not use *bound* variables and is based on Modal Logic S5. $\mathcal{MHC}$ formulas are closer to natural language sentences than formulas using other systems since the order of appearance of properties is similar to their original order in the sentences. $\mathcal{MHC}$ quantifiers can enclose more than one property as this will also follow the same order of properties and quantification in the natural language sentence.

In this dissertation, we proved the *Completeness* of the Monadic Hybrid Calculus $\mathcal{MHC}$. Our proof is based on the Beth-style tableaux. We also proved the *Soundness* of $\mathcal{MHC}$ with respect to the same tableau rules. Furthermore, we proved the equivalence of the Monadic Hybrid Calculus $\mathcal{MHC}$ and the Monadic Predicate Calculus $\mathcal{MPC}$ by giving two algorithms to transform formulas back and forth between the two systems. The *Compactness* proof follows immediately from this result since every formula in $\mathcal{MHC}$ has been proven to have a logically equivalent formula in $\mathcal{MPC}$.

We expanded our scope to include the general Hybrid Predicate Calculus $\mathcal{HPC}$ system that deals with formulas with more than one argument. We explain its novel notations and its formal syntax and formal semantics as well as its pragmatics.

7.2. Future Work

The *Completeness* and *Soundness* proofs presented in this dissertation are for *finite* sets of formulas. Continuing our path to obtain proofs for the Completeness and Soundness for *infinite* sets of formulas will take us too far afield. It involves infinite tableaus that

have infinite sub-trees and infinite paths. However, this issue could be, potentially, one of the future works to improve our results.

Furthermore, the *Compactness* proof already presented is a model theory proof. Another potential idea for the future work is to give a theoretical proof for the Compactness of the Monadic Hybrid Calculus $\mathcal{MHC}$. This work may include giving proofs for both *finite* and *infinite* tableaus.

On the other hand, the future direction of this work could be completely focused on the general Hybrid Predicate Calculus $\mathcal{HPC}$ since we only addressed the Monadic Hybrid Calculus $\mathcal{MHC}$. In other words, a future work could focus on producing the same results found so far for the Monadic Hybrid Calculus $\mathcal{MHC}$ but for formulas that have more than one argument. The work may include the examination of the pragmatic usage of the $\mathcal{HPC}$ system in terms of formalizing natural language sentences.

Moreover, one of the important issues that the future work could focus on is establishing the Tableau Rules (Inference Rules) for the $\mathcal{HPC}$ system that are also based on Beth-style tableaux method. This step is anticipated to be very significant as a longer list of Tableau Rules is expected for the general Hybrid Predicate Calculus $\mathcal{HPC}$ that include more than one argument. Thereafter, the research will proceed to give the proof of the *Completeness, Soundness* and *Compactness* of the entire system of the Hybrid Predicate Calculus $\mathcal{HPC}$.

Finally, a potential future work may include giving algorithms to transform Predicate Calculus $\mathcal{PC}$ formulas to Hybrid Predicate Calculus $\mathcal{HPC}$ formulas and vice versa. These algorithms are anticipated to be much complex than those introduced to transform back and forth between the Monadic Predicate Calculus $\mathcal{MPC}$ and the Monadic Hybrid Calculus $\mathcal{MHC}$.

Bibliography

[1] J. Hintikka. "Semantic Entailment and Formal Derivability by E. W. Beth" in *The Philosophy of Mathematics"*, London: Oxford University Press 1969, pp. 9-41.

[2] W. Newton-Smith. *Logic: An Introductory Course.* London: Routledge & Kegan Paul plc. ,1985, pp.205-208.

[3] R. Feys. "System 5", *Modal Logic.* Belgiumt: The Fondation Universitaire De Belgique, 1965, pp.115-121

[4] T. Brauner. "Hybrid Logic and its Proof-Theory". Applied Logic Series, vol.37, pp. 1-7, 2011.

[5] M. Bergmann, J. Moor, J. Nelson. *The Logic Book.* USA: McGraw-Hill Com.,1998, pp.254-292.

[6] H. Delong. *A Profile of Mathematical Logic.* USA: Addison-Wesley Publishing Com.,1970, pp.113.

[7] W. Newton-Smith. *Logic: An Introductory Course.* London: Routledge & Kegan Paul plc. ,1985, pp.132 & pp.137.

[8] F. Harrison. *Logic and Rational Thought.* USA: West Publishing Com.,1992, pp.347 & pp.351-352.

[9] N. Kretzmann. *Elements of Formal Logic.* USA: The Bobbs-Merrill Com.,1965, pp.117.

[10] H. Kahane. *Logic and Philosophy: A Modern Introduction.* Belmont, CA: Wadsworth, Inc,1990, pp.123.

[11] F. Harrison. *Deductive Logic And Descriptive Language.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1969, pp.323-336.

[12] L. Durst. *The Grammar of Mathematics.* USA: Addison-Wesley Publishing Com.,1969, pp.97.

[13] R. Exner, M. Rosskopf. *Logic In Elementary Mathematics.* USA: McGraw-Hill Book Com.,1959, pp.149.

[14] W. Wadge. "A Hybrid Predicate Calculus". Presented to the Hybrid Logics Workshop, Birmingham, August 2000.

[15] R. Smullyan. *First Order Logic*. New York, USA: Dover Publication, Inc.1995.

[16] S. Reeves and M. Clarke. *Logic for Computer Science*. England: Addison-Wesely Publishing Company, Inc., 1990.

[17] G. Hughes and D. Londey. *The elements of formal logic.* England: Billing and Sons Ltd. 1965.

[18] W. Rautenberg. "First-Order Logic" in *A Concise Introduction to Mathematical Logic*, Springer Science+Business Media, 2010, pp. 41-90.

[19] D. Bühler and W. Minker, "First-Order Logic" in *Domain-Level Reasoning for Spoken Dialogue Systems*, Springer Science+Business Media, 2011. pp. 185-203.

[20] M. Malamud and L. Oridoroga. "Completeness Theorems for Systems of Differential Equations". *Functional Analysis and Its Applications*, Vol. 34, pp.308 -310, 2000.

[21] A. Sinclair. *The Traditional Formal Logic.* London: Methuen & Co. Ltd., 1951, pp. 26-31.

[22] P. Shaw. *Logic and Its Limits.* London: Pan Books Ltd., 1981, pp.32-48.

[23] L. Palmieri. *Language and Clear thinking.* Lincoln: Johnsen Publishing Company, 1960, pp.43-52.

[24] R. Epstein. *The Semantic Foundation of Logic: Predicate Logic.* New York: Oxford University Press, Inc.,1994, pp.109-114 and 304-352.

[25] H. Kahane. *Logic and Philosophy: A Modern Introduction.* Belmont, CA: Wadsworth, Inc,1990, pp.40-41.

[26] R. Girle. *Possible Worlds.* Montreal and Kingston: McGill-Queen's University Press, 2003, pp.38-40, 146-150.

[27] R. Hughes. *A Philosophical Companion to First-Order Logic.* Indianapolis, Indiana: Hackett Publishing Company, Inc., 1993, pp.85-93

[28] S. Guttenplan. *The Language of Logic.* Oxford, UK: Basil Blackwell Ltd. 1986, pp. 293-295.

[29] R. Jeffrey. *Formal Logic: Its Scope And Limits.* USA: McGraw-Hill, Inc., 1967, pp.55-56. pp.159-168.

[30] F. Harrison. *Deductive Logic And Descriptive Language.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1969, pp.287-290.

[31] J. Garsom. "Modal Logic". Internet: http://plato.stanford.edu/entries/logic-modal/ , May 27, 2014 [Feb 28, 2015].

[32] H. Kahane. "Modal, Epistemic, and Deontic Logic ", *Logic and Philosophy: A Modern Introduction.* Belmont, CA: Wadsworth, Inc,1990, pp.407-417.