Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

On the Descriptive Complexity of Satisfiability on Quantified Boolean Formulas

A thesis presented in partial fulfilment of

the requirements for the degree of

Master

Of

Information Sciences

At Massey University,

Wellington,

New Zealand

Wei Ren (02103176)

Mar 2011

Thesis Supervisor: Dr. Jose Maria Turull-Torres

Abstract

In the present thesis, we deal with the construction of non trivial formulas in higher order logic languages. In particular, we focus on using SO (Second-Order logic) and TO (Third-Order logic) to express $SATQBF_k$, and $SATQBF$ respectively. First of all, we explain the relationship between logic languages and complexity classes. Then we give formal definitions and examples for FO (First-Order), SO and HO^i $(i \geq 2)$ (Higher-Order logic). It is known that, for every $k \geq 1$, $SATQBF_k$ is a complete problem for the level Σ_k^P of PH (Polynomial-time hierarchy), and that $SATQBF$ is a complete problem for PSPACE. As the expressibility of SO is known to equal the class PH, then we know that there must be an SO formula which can express $SATQBF_k$. On the other hand, PSPACE is known to be equal in expressive power to SO with the addition of a second order transitive closure quantifier, which is widely conjectured to be strictly more expressive than SO alone. As TO includes $PSPACE$, this means that there must be a TO formula that can express $SATQBF$. Here we give first a top down explanation on the use of SO to express $SATQBF_k$. A detailed SO formula is presented. We then give a top down presentation of the sketch of a TO formula for $SATQBF$.

Acknowledgments

I am deeply grateful to my supervisor, Dr. Jose Maria Turull-Torres. Since the beginning, he has been teaching and sharing his knowledge with me, and always supported and encouraged me. I have to thank my lovely wife Xue as well, who has always supported me. Also big thanks to my former teacher and friend Dr. Keri Logan, who did the proofreading. One more person that I have to say thanks is my company manager Jeremy, who has given me great support and suggestions when I have had difficulty handling work and study at the same time. I am also very grateful to the two reviewers of my thesis, Dr. C. McCartin and Dr. F. Ferrarotti, for their careful reading and their comments and suggestions. Without all their help and supports, I would not have been able to complete my master thesis on time.

Contents

1 Introduction

For many years there has been a lot of interest in query languages for relational databases. As we know query languages are specialized languages for asking questions, or queries, which involve the data in a database [RG03]. There are two major types of relational query languages, logical and algebraic. Logical languages, such as the language of relational calculus contain formulas which when evaluated on a database return as an answer the set of all tuples that satisfy them. These types of languages are non-procedural in nature. Algebraic languages, which contain relational algebra, consist of programs whose basic operations are algebraic ones like join and projection. These types of languages are procedural in nature [Var82].

Between these two types of query languages, strong relationships exist. Many researchers have been working on this subject. Therefore, how to distinguish these two types of query languages, and what the differences are, are important questions that we have to face. There are different ways to measure the complexity of evaluating queries in a specific language. One of the ways is to compare the expressive power of these languages. Computational complexity theory is brought into this field; it measures the amount of computational resources, such as time and space that are needed, as a function of the size of the input, to compute a query [Imm99]. It helps us to compare the expressibility of different query languages by looking at their corresponding complexity classes.

Regarding relational database theory, there is a lot of information that is related. Following Codd's pioneering work [Cod72] on relational calculus and relational algebra, a lot of work has been done studying and comparing the expressive power of these query languages. A central topic in Database Theory in recent years is to study the expressive power of different logics which are built as different sorts of extensions to first-order logic (FO), or equivalently, relational calculus or relational algebra, used

as computation models for the expression of queries to relational databases, which can be sketched in the following graph (See Figure 1).

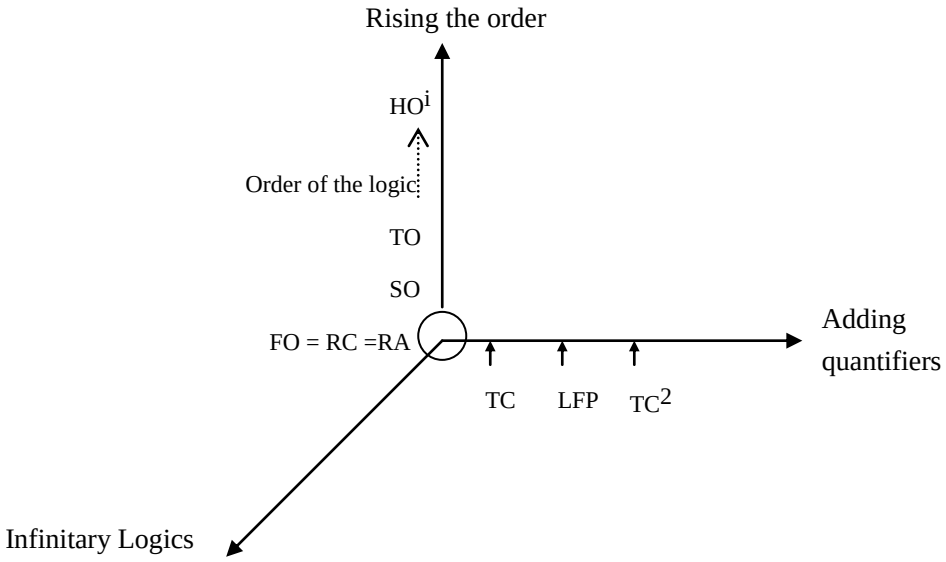


Figure 1: Expressibility of different logic languages

Looking at the graph above, in a more pure logic setting, the problem is regarded as the expressibility of different logics in finite model theory **[**EF99**]**. Finite model theory studies, among other subjects, the expressive power of logics on finite models, in contrast to classical model theory, on the other hand, which concentrates on infinite structures, given that most subjects of interest in mathematics deal with infinite structures. There are many interesting research areas in finite model theory. One is interested in the expressiveness of logics over finite graphs, finite strings, or other finite relational structures.

However it seems that not much attention has been devoted in the literature to the actual use of higher order logics to express non trivial properties of structures. With our work in [RT08] and in this thesis we aim to start a line of research in that direction.

In the present work by *QBF* (Quantified Boolean Formulas) we mean the formulas of

the form $F = Q_1 x_1 Q_2 x_2 ... Q_n x_n (E)$, where E is a Boolean expression involving the variables $x_1, x_2, ..., x_n$ and each Q_i is either "$\exists$" or "$\forall$". For $k \geq 1$, by QBF_k we mean the QBF formulas which have k alternated blocks of quantifiers, starting with an existential block. $SATQBF_k$ and $SATQBF$ denote respectively, the problems of deciding whether a QBF_k formula or a QBF formula is satisfiable. As we know, for every $k \geq 1$, $SATQBF_k$ is Σ_k^P complete, where $\Sigma_k^P = NP^{NP^{NP^{\cdot^{\cdot^{\cdot}}}}}$. We also know that $SATQBF$ is a complete problem for PSPACE. As according to computational complexity theory, the expressibility of SO (Second-Order logic) equals the class PH (Polynomial-time hierarchy), then we know that, for every $k \geq 1$, $SATQBF_k$ can be expressed in SO. On the other hand, PSPACE is known to be equal in expressive power to SO with the addition of a second order transitive closure operator, which is widely conjectured to be strictly more expressive than SO alone. As TO includes $PSPACE$, this means that there must be a TO formula that can express $SATQBF$. However, it seems that how to express these problems in SO and TO, respectively, is far from well known. In this thesis, we give in full detail an SO formula to express $SATQBF_k$ and then we give a sketch of a TO formula to express $SATQBF$.

In Section 2, we will give examples of graph properties expressed in different logics. We will mainly focus on FO, SO, TO and the expressive power that each of them have through a few examples of queries. Once we have a basic picture about what they are, we will focus in the following sections on the expressibility of second order logic, using this logic to express $SATQBF_k$. And then we will give in Subsection 5.3 a top down presentation of the sketch of a TO formula for $SATQBF$.

2 Background

2.1 Finite Model Theory, Relational Database and Queries

Finite Model Theory and Relational Database Theory are two disciplines intimately connected. While Finite Model Theory provides a solid theoretical foundation to databases, databases provide one of the main concrete scenarios for Finite Model Theory within computer science. And most of the overlaps between Finite Model Theory and Relational Database Theory occur in the theory of query languages. Chandra and Harel **[**CH80**]** formalized the notion of computable queries for relational databases in 1980, and since then intensive studies have been done along this direction. There is much research which has focused on the completeness of different query languages and their expressive powers or computational complexities.
From a theoretical perspective, it is desirable for a model of computation of queries to be representation independent, which means queries to databases that represent the same reality should evaluate to the same result. In mathematical terms, Chandra and Harel **[**CH80**]** partially captured the previous concept by asking queries to isomorphic databases to evaluate to the same result. Therefore, the subject of query languages has become the most important bridge between Finite Model theory and relational databases, since logic languages are one of the ways to express queries.

2.2 Relationship with Complexity classes

Computational complexity theory and logic languages are two fundamental areas in computer science. In computational complexity theory there are several basic notions and ongoing questions of which we have to be aware. Such as different computational models, some classic complexity classes, the relationship between different complexity classes, how to classify them, and then, what relationships between these complexity classes have been discovered, and what have not. And why we need to

define these complexity classes. Let us look at the different perspectives.

Before we go any further, let us look at an overall picture of the relationship between different classic complexity classes and logics: (***Figure 2***)

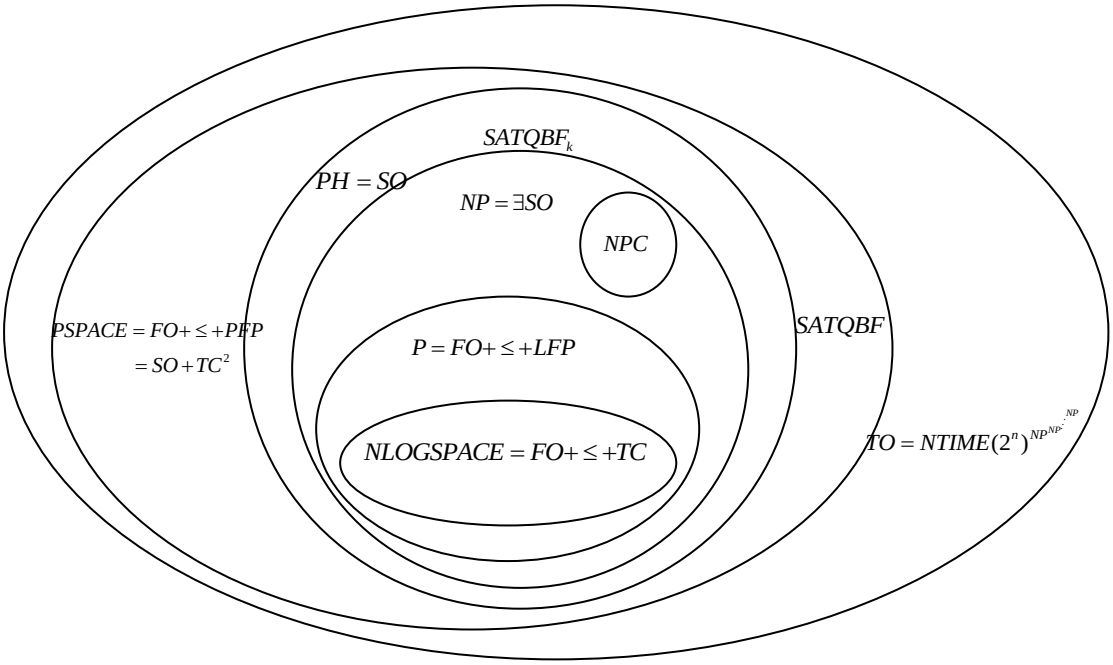


Figure 2: Relationship between different classic complexity classes and logics

From the graph above, we can clearly see the relationship among these complexity classes. Most researchers believe it looks like the graph above, i.e., that all inclusions are proper. The known relationships are, however, as follows, where $\subset$ means strict inclusion, and $\subseteq$ means inclusion.

$$\mathrm{NLOGSPACE} \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PH} \subseteq \mathrm{PSPACE} \subseteq \mathrm{DTIME}(2^{n^{O(1)}}) \subseteq \mathrm{NTIME}(2^{n^{O(1)}})$$

$$\mathrm{P} \subset \mathrm{PSPACE}$$

For instance, we know that NLOGSPACE is included in P (PTIME), but we still do not know if there are any problems in P, which are not in NLOGSPACE. This means that it could be the case that P = NLOGSPACE. The following are known relationships:

NLOGSPACE $\subset$ PSPACE

P $\subset$ EXPTIME = DTIME $(2^{n^{O(1)}})$

2.3 Using Logic to Express Database Queries

2.3.1 First-Order logic (FO)

Following [TT03], a First Order language L (σ) is the set of formulas built up from the relation symbols of σ and the relation symbols: =, $\leq$ using logical connectives: $\vee, \wedge, \neg$ variables: x, y, z, …, and quantifiers: $\forall, \exists$. The relation symbol $\leq$ refers to a total ordering on the domain of the structure, and the quantifiers range over this domain **(**Universal quantifier $\forall$ for All, Existential quantifier $\exists$ for Exists**)**. A term is either a variable or a constant symbol in the schema.

A schema or vocabulary is as follows: $\sigma = < R_1, ..., R_s >$, where for $1 \leq i \leq s$, R_i is a relation symbol of arity $r_i \geq 1$.

Let σ be a schema, we define the set of atomic formulae on the schema σ as follows:

1. If R is a relation symbol in σ of arity r, for some $r \geq 1$, and $t_0, ..., t_{r-1}$ are variables, then $R(t_0, ..., t_{r-1})$ is an atomic formula.

2. If t_0 and t_1 are variables, then $t_0 = t_1$ is an atomic formula.

3. Nothing else is an atomic formula.

We define the set of well formed formulae as follows:

1. An atomic formula is a well formed formula.

2. If φ, ψ are well formed formulae, then the following are also well formed formulae: $(\neg\varphi), (\varphi \wedge \psi), (\varphi \vee \psi), (\varphi \rightarrow \psi)$

3. If φ is a well formed formulae, and x is an individual variable, then the following are also well formed formulae: $\exists x(\varphi), \forall x(\varphi)$

4. Nothing else is a well formed formula.

The variable x is said to be bound in the formulae $\exists x(\varphi)$ and $\forall x(\varphi)$. If a variable is not bound in a formula, then it is said to be free in that formula. By $\varphi(x_1, ..., x_r)$ we denote a formula of First-Order logic whose free variables are exactly $\{x_1, ..., x_r\}$. A sentence is a formula with no free variables.

How to use FO in a particular context, such as relational databases, becomes an important issue. Here we need to define the semantics for relational calculus, which is also considered to have the same expressive power as First-Order Logic. There are two basic concepts.

A structure (or database instance) of schema σ is as follows: $A=<D^A, R_1^A, ..., R_s^A>$, where D^A is a finite set, for $1 \le i \le s$, R_i^A is a relation of arity r_i on D^A which interprets the relation symbol R_i in σ.

Firstly, assume we have a relational schema σ. A valuation v on a database A of

schema σ, is a function which assigns to each individual variable x an element in dom(A).

Let v_0, v_1 be two valuations on a database A of schema σ, and let x be a variable, we say that v_0 and v_1 are x-equivalent if they coincide in every variable, with the possible exception of variable x.

Secondly, considering the same database A of schema σ, if we have a valuation v on the database A, we can define inductively the notion of satisfaction, which is denoted as |= .

A, v |= R ($x_0, ..., x_{r-1}$) where R is a relation symbol in σ of arity r, for some $r \geq 1$, and $x_0, ..., x_{r-1}$ are individual variables, if and only if the r-tuple (v(x_0), …, v(x_{r-1})) belongs to the relation R^A.

A, v |= $t_0 = t_1$ where t_0, t_1 are variables, if and only if v (t_0) = v (t_1).

A, v|= $\neg\varphi$ where φ is a well formed formula, if and only if it is not the case that A, v |= φ.

A, v|= $\varphi \wedge \psi$ where φ, ψ are well formed formulae, if and only if A, v|= φ and A, v|= ψ.

A, v|= $\varphi \vee \psi$ where φ, ψ are well formed formulae, if and only if A, v|= φ or A, v|= ψ, or both hold.

A, v|= $\exists x(\varphi)$ where x is an individual variable and φ is a well formed formula, if and only if there is at least one valuation v', which is x-equivalent to v, such that A, v'|= φ. That is, if and only if there is at least one element in the domain of the database, such that A, v|= φ when that element replaces the variable x in the formula φ.

A, v|= $\forall x(\varphi)$ where x is an individual variable and φ is a well formed formula, if and only if for every valuation v', which is x-equivalent to v, A, v'|= φ. That is, A, v|= φ if we replace the variable x in the formula φ with any element in the domain of the database.

Let σ be a schema, let r $\geq$ 1, and let R be a relation symbol of arity r. A query of arity r and schema σ is a function q: $B_\sigma \rightarrow B_{\langle R \rangle}$ that preserves isomorphisms, and such that for every database A of schema (σ), dom (q (A)) $\subseteq$ dom (A). That is, all the elements which form the output to the query q when evaluated on a given database A must belong to the domain of that database.

A Boolean query is a function q: $B_\sigma \rightarrow$ {TRUE,FALSE} that preserves isomorphisms.

If $\varphi(x_1,...,x_k)$ is a formula of schema σ with free variable { $x_1,...,x_k$ }, A is a database of schema σ, and $a_1,...,a_k$ are elements of the domain of A, with A|= $\varphi(x_1,...,x_k)$ [$a_1,...,a_k$], we denote that φ is TRUE, when interpreted by A, under a valuation v where for 1 $\leq$ i $\leq$ k it is v (x_i) = a_i. Now we consider the set of all such valuations as follows:

$$\varphi^A = \{(a_1,...,a_k):\ a_1,...,a_k \in \text{dom (A)} \ \wedge\ \text{A}|= \varphi(x_1,...,x_k)[a_1,...,a_k]\}$$

That is, φ^A is the relation defined by φ in the database A, and its arity is given by the number of free variables in φ. Formally, we say that a formula $\varphi(x_1,...,x_k)$ of schema σ, expresses a query q of schema σ, if for every database A of schema σ, is q(A) = φ^A. Similarly, a sentence φ expresses a Boolean query q if for every database A of schema σ, is q(A) = TRUE if and only if A|= φ.

2.3.1.1 Examples

We will denote the input degree of a node in a directed graph as id and the output degree as od.

We consider a database of schema $\sigma =< F >$, where F has the following semantics:

Departure City (From)	Arrival City (To)

Query 1: "Get the cities from which there are flights to exactly one city" which means that their output degree is 1.

To express this in FO logic: $\phi(x) = \exists y(F(x,y) \wedge \forall z(F(x,z) \rightarrow (z = y))$, which means that there is one city y, such that there is a flight from x to y, and that for every city z, if there is a flight from x to that city z, then z is equal to y. x in this formula is the only free variable, because it does not have any quantifier which binds it. We use $\phi(x)$ to denote this formula. Consider a simple database instance as in (***Figure 3***); the result

for the query will be the unary relation {(PN)}.

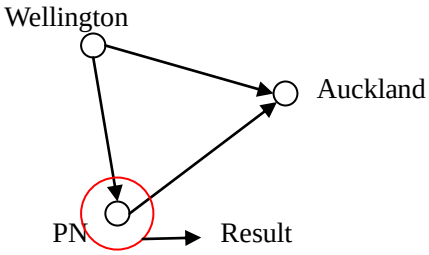


Figure 3

Query 2: "Get the cities from which there are flights to exactly 2 other cities" (output degree =2).

Suppose that a given database contains the name of cities and pairs (a, b) of such cities such that a given airline offers services from A to B without stopover. Let $\sigma =< F^2 >$ be the database schema, and let J1 be a database instance of σ:

From	To
A	D
A	B
B	C
D	E
B	F

$J_1 = \{ D^{J_1}, F^{J_1} \}$ where D^{J_1} = {A, B, C, D, E, F}, and

F^{J_1} = {(A, D), (A, B), (B, C), (D, E), (B, F)}

J_1:

This query in logic:

$$\phi(x) = \exists z_1 z_2 (z_1 \neq z_2 \wedge F(x, z_1) \wedge F(x, z_2) \wedge \forall z_3 (F(x, z_3) \rightarrow (z_3 = z_1 \vee z_3 = z_2)))$$

*Query 3: Get the cities with flights to all cities from which there is exactly one flight***.**

$$\phi(x) = \forall z_1(\underbrace{\exists z_2(F(z_1, z_2) \wedge \forall z_3(F(z_1, z_3) \rightarrow (z_3 = z_2))}_{\text{This part expresses that the output degree of } z_1 \text{ is 1. It will give us all the cities whose output degree is 1.}} \rightarrow F(x, z_1))$$

Once we have all the cities with output degree 1, we say that for all those cities there is a flight from x to them.

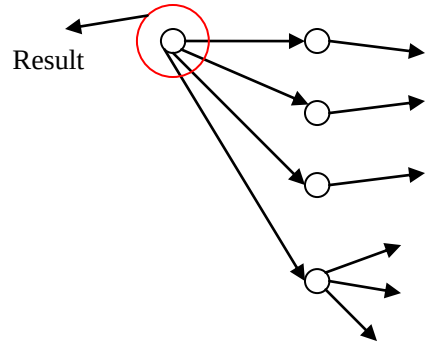


Figure 4

If we slightly change the query, as follows: *Get the cities with flights to EXACTLY all cities from which there is exactly one flight*, and then the result will be different:

Not in the Result. Because this city has a flight to a city whose output degree is greater than 1.

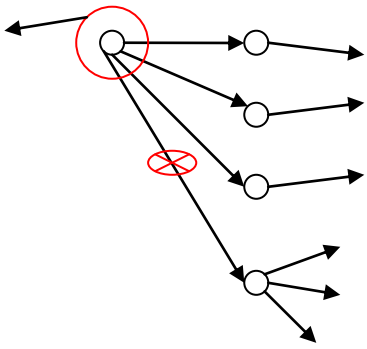


Figure 5

2.3.2 Second-Order logic

Second-Order Logic is an extension of First Order Logic, which allows quantifying over relations.

Definition 1 (Second-order logic) **([**Lib04**],** p113**)**. The definition of second-order logic, SO, extends the definition of FO with second-order variables, ranging over subsets and relations on the universe, and quantification over such variables. We assume that for every k > 0, there are infinitely many variables $X_1^k, X_2^k, ...,$ ranging over k-ary relations. A formula of SO can have both first-order and second-order free variables; we write $\varphi(\vec{x}, \vec{X})$ to indicate that $\vec{x}$ are free first-order variables, and $\vec{X}$ are free second-order variables.

Given a vocabulary or schema σ that consists of relation and constant symbols, we define SO terms and formulae, and their free variables, as follows:

1. Every first-order variable x, is a first-order term. The only free variable of a term x is variable x.

2. There are three kinds of atomic formulae:

   a) FO atomic formulae; that is, formulae of the form
      i. t = t', where t, t' are variables, and
      ii. R($\vec{t}$), where $\vec{t}$ is a tuple of variables, and $R \in \sigma$, and

   b) X ($t_1, ..., t_k$), where $t_1, ..., t_k$ are variables, and X is a second-order variable of arity k. The free first-order variables of this formula are free first-order variables of $t_1, ..., t_k$; the free second-order variable is X.

3. The formulae of SO are closed under the Boolean connectives $\vee, \wedge, \neg$ and first-order quantification, with the usual rules of free variables.

4. If $\varphi(\vec{x}, Y, \vec{X})$ is a formula, then $\exists Y\ \varphi(\vec{x}, Y, \vec{X})$ and $\forall Y\ \varphi(\vec{x}, Y, \vec{X})$ are formulae, whose free variables are $\vec{x}$ and $\vec{X}$.

The semantics is defined as follows. Suppose A is a database of schema σ. For each formula $\varphi(\vec{x}, \vec{X})$, we define the notion of satisfaction $A \models \varphi(\vec{b}, \vec{B})$, where $\vec{b}$ is a tuple of elements of D^A of the same length as $\vec{x}$, and for $\vec{X} = (X_1, ..., X_l)$, with each X_i being of arity n_i, $\vec{B} = (B_1, ..., B_l)$, where each B_i is subset of $(D^A)^{n_i}$.
We give the semantics only for constructions that are different from those for FO:

1. If $\varphi(\vec{x}, \vec{X})$ is X $(t_1, ..., t_k)$, where X is k-ary and $t_1, ..., t_k$ are terms, with free variables among $\vec{x}$, then $A \models \varphi(\vec{b}, B)$ if and only if the tuple $(t_1^A(\vec{b}), ..., t_k^A(\vec{b}))$ is in B.

2. If $\varphi(\vec{x}, \vec{X})$ is $\exists Y\ \varphi(\vec{x}, Y, \vec{X})$, where Y is k-ary, then $A \models \varphi(\vec{b}, \vec{B})$ if for some $C \subseteq (D^A)^k$, it is the case that $A \models \varphi(\vec{b}, C, \vec{B})$.

3. If $\varphi(\vec{x}, \vec{X})$ is $\forall Y\ \varphi(\vec{x}, Y, \vec{X})$, and Y is k-ary, then $A \models \varphi(\vec{b}, \vec{B})$ if for all $C \subseteq (D^A)^k$, we have $A \models \varphi(\vec{b}, C, \vec{B})$.

Definition 2 ([Lib04**], p115)** Existential SO logic, or $\exists SO$, is defined as the restriction of SO that consists of the formulae of the form $\exists X_1 ... \exists X_n \varphi$, where φ does not have any second-order quantification. If, furthermore, all X_i's have arity 1, the resulting restriction is called existential monadic SO, or $\exists MSO$. If the second-order quantifier prefix consists only of universal quantifiers, we speak of the universal SO logic, or $\forall SO$, and its further restriction to monadic quantifiers is referred to as $\forall MSO$.

2.3.2.1 Examples

We include below some examples from **[**FT04**]**, among other sources.

Here is an example of how to use Second-Order Logic to express a query that represents a well known problem, which is known in graph theory as "3–colorability". Let us consider a graph G with the schema $\sigma =< E^2 >$. We have to check whether the graph G is 3 colorable, which means that the nodes in G can be colored, using three different colors, in such a way that each node has exactly one color and no two nodes with the same color are connected by an edge. The query can be expressed by an existential Second-Order formula of the form: $\exists RYB(\phi)$, where R, Y, B are unary predicates (sets that represent the 3 colors, Red(R), Yellow(Y), and Blue(B)), and ϕ is a First-Order formula.

Note that "3-colorability" can be used in the flights example. Here is the query: *"Can we split the set of cities in 3 groups, in such a way that no direct flight connects two cities of the same group?"*

Departure City (From)	Arrival City (To)

$\phi \equiv \exists$RBY("all cities belong exactly to 1 set, R, B or Y, and whenever we have a flight it connects two cities which belong to different sets")

$$\begin{aligned}\phi \equiv \exists \text{RBY}(&\forall x((\text{R}(x) \vee \text{Y}(x) \vee \text{B}(x)) && \text{"all nodes have a colour"}\\ &\wedge \text{R}(x) \rightarrow (\neg \text{Y}(x) \wedge \neg \text{B}(x))\\ &\wedge \text{Y}(x) \rightarrow (\neg \text{R}(x) \wedge \neg \text{B}(x))\\ &\wedge \text{B}(x) \rightarrow (\neg \text{R}(x) \wedge \neg \text{Y}(x)) && \text{"1 single colour per node"}\end{aligned}$$

$$\wedge \forall y(\text{Flights}(x,y) \rightarrow \neg(R(x) \wedge R(y)) \quad \text{"all edges link two nodes of}$$
$$\wedge \neg(Y(x) \wedge Y(y)) \qquad \text{different colors"}$$
$$\wedge \neg(B(x) \wedge B(y)))))$$

Second-Order logic differs from First-Order logic in that it has variables and quantifiers not only for individuals but also for n-ary relations. The expressive power of Second-Order logic is higher than First-Order logic; for instance, the transitive closure query and 3-colorability (an NP complete problem) are not expressible in First-Order logic, but can be expressed in Second-Order logic.

Query: *"Which pairs of cities are connected through flights, either with or without stopovers?"* This is the transitive closure or reachability query, and it uses binary existential Second-Order logic (bin∃SO).

$\phi(v,w) \equiv \exists S^1 R^2$ ("the graph(S,R) is a subgraph of Flights, and one of its connected componets is linear, with root v and leaf w")

$$\phi(v,w) \equiv \exists S \exists R(S(v) \wedge S(w) \qquad \text{"v,w} \in \text{S"}$$

$$\wedge \forall xy(R(x,y) \rightarrow F(x,y)) \qquad \text{"R} \subseteq \text{Flights"}$$

$$\wedge \forall xy(R(x,y) \rightarrow S(x) \wedge S(y)) \qquad \text{"R} \subseteq S \times S\text{"}$$

$$\wedge \forall x(S(x) \rightarrow \exists y(R(x,y) \vee R(y,x)))$$
"there are no isolated nodes in the graph (S,R)"

$$\wedge \forall x(S(x) \rightarrow x=w \wedge \neg\exists y(R(x,y))$$
$$\vee$$
$$x \neq w \wedge \exists y(R(x,y) \wedge \forall z(R(x,z) \rightarrow y=z)))$$
"w is a leaf, and all other nodes have od=1"

$$\wedge \forall x(S(x) \rightarrow x=v \wedge \neg\exists y(R(y,x))$$
$$\vee$$
$$x \neq v \wedge \exists y(R(y,x) \wedge \forall z(R(z,x) \rightarrow y=z)))$$
"v is a root, and all other nodes have id=1"

2.3.3 Third Order logic (TO)

In the alphabet of third order logic besides the usual logical and punctuation symbols, we have a countable infinite set of individual variables, for every arity, a countable infinite set of second order relation variables, and for every arity and every type, a countable infinite set of relation variables of third order. A type is a particular combination of individual and SO relation variables. We use upper case letters like X and Y for relation variables, and lower case letters like x and y for individual variables.

Let σ be a schema or relational vocabulary. We define the set of atomic formulae on the vocabulary σ as follows:

(1) If R is a relation symbol in σ of arity r, for some $r \geq 1$, and $x_0, ..., x_{r-1}$ are individual variables, then R ($x_0, ..., x_{r-1}$) is an atomic formula.

(2) If x and y are individual variables, then x = y is an atomic formula.

(3) If X is a relation variable of order 2, and of arity r, for some $r \geq 1$, and $x_0, ..., x_{r-1}$ are individual variables, then X ($x_0, ..., x_{r-1}$) is an atomic formula.

(4) If X is a relation variable of order 3 and of arity r, for some $r > 0$, and for $0 \leq i < r$, Y_i is either an individual variable or a relation variable of order 2, then X $(Y_0, ..., Y_{r-1})$ is an atomic formula, provided that at least one of the variables Y_i is a relation variable of order 2, The particular tuple of individual and relation variables must agree with the type of the third order variable X.

(5) Nothing else is an atomic formula.

Let $r > 0$. A second-order relation of arity r is a relation in the classical sense, i.e., a set of r-tuples of elements of the domain of a given structure. A relation of order 3 of arity r, or a third order relation of arity r, is a set of r-tuples where each component is either a relation of order 2, or an element of the domain of a given structure, according to the type of the third order relation.

Let σ be a relational vocabulary. A valuation v on a σ-structure A, is a function which assigns to each individual variable x an element in dom(A), and to each relation variable X of order j, for some $1 < j < 4$, and of arity r, for some $r > 0$, a relation of order j, arity r, and of the same type as X if j=3, on dom(A). Let v_0, v_1 be two valuations on a σ-structure A, and let V be a variable of whichever kind, we say that v_0 and v_1 are V-equivalent if they coincide in every variable of whichever kind, with the possible exception of variable V. We also use the notion of equivalence w.r.t. sets of variables. Let A be a σ-structure, and let v be a valuation on A. Next, we define inductively the notion of satisfaction in TO.

(1) $A, v \models R(x_0, ..., x_{r-1})$, where R is a relation symbol in σ of arity r, for some $r \geq 1$, and $x_0, ..., x_{r-1}$ are individual variables, iff the r-tuple $(v(x_0), ..., v(x_{r-1}))$ belongs to the (second-order) relation R^A.

(2) A, $v \models X(x_0, ..., x_{r-1})$, where X is a relation variable of order 2 and of arity r, for some $r \geq 1$, and $x_0, ..., x_{r-1}$ are individual variables, iff the r-tuple $(v(x_0), ..., v(x_{r-1}))$ belongs to the second-order relation v(X).

(3) $A, v \models X(Y_0, ..., Y_{r-1})$, where X is a relation variable of order 3, and of arity r, for some $r > 0$, and for $0 \leq i < r$, Y_i is either an individual variable or a relation variable

of order 2, iff the r-tuple of elements and/or relations of order 2 (according to the type of X), $(v(Y_0),...,v(Y_{r-1}))$ belongs to the relation of order 3 v(X).

(4) A, $v \models x = y$, where x, y are individual variables, iff v(x) = v(y).

(5) A, $v \models \neg\varphi$, where φ is a well-formed formula, iff it is not the case that A, $v \models \varphi$.

(6) A, $v \models \varphi \wedge \psi$, where φ, ψ are well-formed formula, iff A, $v \models \varphi$ and A, $v \models \psi$

(7) A, $v \models \varphi \vee \psi$, where φ, ψ are well-formed formula, iff either A, $v \models \varphi$ or A, $v \models \psi$, or both hold.

(8) A, $v \models \exists x(\varphi)$, where x is an individual variable and φ is a well-formed formula, iff there a valuation v', which is x-equivalent to v, such that A, $v' \models \varphi$

(9) A, $v \models \forall x(\varphi)$, where x is an individual variable and φ is a well-formed formula, iff for every valuation v', which is x-equivalent to v, A, $v' \models \varphi$

(10) A, $v \models \exists X(\varphi)$, where X is a relation variable, and φ is a well-formed formula, iff there is a valuation v', which is X-equivalent to v, such that A, $v' \models \varphi$

(11) A, $v \models \forall X(\varphi)$, where X is a relation variable, and φ is a well-formed formula, iff for every valuation v', which is X-equivalent to v, A, $v' \models \varphi$

2.3.3.1 Examples in Third-Order Logic (TO)

We include below some examples from [FT04]. Let's consider the schema $\sigma = \langle \text{Bought}^2 \rangle$ of the sales database together with its intuitive meaning. That is, if R is a relation instance of relation schema Bought = (Customer, Article), a pair $(x,y) \in R$ iff customer x bought article y. Here $S^{3,1}$ is a relation variable of order 3 and arity 1, and $R^{3,2}$ is a relation variable of order 3 and arity 2.

$\phi \equiv \exists S^{3,1}(\forall s^{2,1}(S^{3,1}(s^{2,1}) \rightarrow \exists x(\forall y(\text{Bought}(x,y) \leftrightarrow s^{2,1}(y))))$
"each set s in S is the set of articles bought by some customer"

$\wedge \forall x(\exists s^{2,1}(S^{3,1}(s^{2,1}) \wedge \forall y(\text{Bought}(x,y) \leftrightarrow s^{2,1}(y))))$
"for each customer, the set of articles bought by him/her, is a set in S"
$\wedge ...)$
$\equiv \exists S^{3,1}(\text{"}S^{3,1}$ is the set of sets of articles bought by the different customers and..."$)$

Then we can express properties of the sets of articles bought by different customers, for example: *"For exactly half of the possible subsets of articles, there is some customer who bought those articles".* The existential Third-Order formula for this query:

$\phi \equiv \exists S_1^{3,1} S_2^{3,1}(\forall S^{2,1}((S_1^{3,1}(S^{2,1}) \rightarrow \forall y(S^{2,1}(y) \rightarrow \exists x(\text{Bought}(x,y))))$
"$S_1^{3,1}$ is a set of subsets of articles"

$\wedge (S_2^{3,1}(S^{2,1}) \rightarrow \forall y(S^{2,1}(y) \rightarrow \exists x(\text{Bought}(x,y))))$
"$S_2^{3,1}$ is a set of subsets of articles"

$\wedge ((\forall y(S^{2,1}(y) \rightarrow \exists x(\text{Bought}(x,y)))) \rightarrow (S_1^{3,1}(S^{2,1}) \vee S_2^{3,1}(S^{2,1})))$
"every subset of articles is either in $S_1^{3,1}$or in $S_2^{3,1}$"

$\wedge(S_1^{3,1}(S^{2,1}) \leftrightarrow \neg S_2^{3,1}(S^{2,1}))$ "no set is in both $S_1^{3,1}$ and $S_2^{3,1}$"

$\wedge(S_1^{3,1}(S^{2,1}) \rightarrow \exists x(\forall y(S^{2,1}(y) \leftrightarrow (\text{Bought}(x,y))))$
"every set in $S_1^{3,1}$ is the set of articles bought by some customer"

$\wedge(S_2^{3,1}(S^{2,1}) \rightarrow \neg\exists x(\forall y(S^{2,1}(y) \leftrightarrow (\text{Bought}(x,y))))$
"no set in $S_2^{3,1}$ is the set of articles bought by some customer"

$\wedge \exists R^{3,2}(\forall S_1^{2,1} S_2^{2,1} S_3^{2,1}((R^{3,2}(S_1,S_2) \rightarrow S_1^{3,1}(S_1) \wedge S_2^{3,1}(S_2))$ "$R \subseteq S_1 \times S_2$"

$\wedge(R^{3,2}(S_1,S_2) \wedge R^{3,2}(S_1,S_3) \rightarrow S_2 = S_3)$ "function"

$\wedge(R^{3,2}(S_1,S_2) \wedge R^{3,2}(S_3,S_2) \rightarrow S_1 = S_3)$ "injectivity"

$\wedge(S_1^{3,1}(S_1) \rightarrow \exists S_2(R^{3,2}(S_1,S_2)))$ "totality"

$\wedge(S_2^{3,1}(S_2) \rightarrow \exists S_1(R^{3,2}(S_1,S_2))))$ "surjectivity"

Note that $R^{3,2}$ is a bijection if and only if it is a function which is total, injective and surjective, and that "there is a bijection between $S_1^{3,1}$ and $S_2^{3,1}$" $\equiv |S_1^{3,1}| = |S_2^{3,1}|$.

3 Sub-formulas and rules used throughout the thesis

In what follows, we will use the following conventions.

1. Variables
   a. Lowercase for first order variables, like v_t.
   b. Uppercase for second order variables, like V_t.
   c. Large Uppercase for third order variables, like $\mathbf{V}_t$.

d. We will use super indices in variables. $V_t^{2,1}$ is a second order variable with arity 1, $E_t^{2,2}$ is a second order variable with arity 2. $V_T^{3,1}$ is a third order variable with arity 1, $E_T^{3,2}$ is a third order variable with arity 2, and so on. When we use only one super-index, we mean an SO variable of that arity.

2. Parenthesis: some parentheses are indexed for matching different levels.

   We will usually index "(" and "[" starting from 0 in the outermost level, like $(_0(_1(_2 \quad)_2)_1)_0$.

3. As disjunctions and conjunctions are associative, we do not use parenthesis to enclose the two operations. That is, we write $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$ instead of $\varphi_1 \wedge (\varphi_2 \wedge \varphi_3)$.

4. We write $\varphi_1 \wedge \neg\varphi_2$ instead of $\varphi_1 \wedge (\neg\varphi_2)$.

5. Throughout this thesis by “graph” we mean “directed graph”.

The following sub-formulas are well known. They can be found in [RT08] among other sources.

(For an edge relation E and total order $\leq$)

1. Successor: $SUC_{\leq}(x,y) \equiv \leq (x,y) \wedge \forall v[(\leq (x,v) \wedge \leq (v,y)) \rightarrow (v = x \vee v = y)]$

2. Predecessor: $PRED_{\leq}(y,x) \equiv SUC_{\leq}(x,y)$

3. Path: $PATH_E(v,w) \equiv$ "$(v,w) \in TC(E)$"; note that $(v,v) \in TC(E)$ (we will also use $PATH_E^{2,4}(x_1,y_1,x_2,y_2)$ where E is a 2nd order relation of arity 4, meaning that there is a path in E from (x_1,y_1) to (x_2,y_2)).

$PATH_E(v,w)$

$\equiv \exists \mathrm{V}''^1 \mathrm{E}''^2 (\forall v', v''(\mathrm{E}''(v',v'') \to \mathrm{V}''(v') \wedge \mathrm{V}''(v'') \wedge \mathrm{E}(v',v'')) \wedge \forall v'(\mathrm{V}''(v') \to \mathrm{V}(v'))$

$\wedge (\mathrm{V}''(v) \wedge \mathrm{V}''(w) \wedge \neg \exists v'(\mathrm{E}''(v',v) \wedge \forall v''[\mathrm{V}''(v'') \wedge (v'' \neq v) \to \exists v'(\mathrm{E}''(v',v''))]$

"v is the only minimal node"

$\wedge \neg \exists v'(\mathrm{E}''(w,v') \wedge \forall v''[\mathrm{V}''(v'') \wedge (v'' \neq w) \to \exists v'(\mathrm{E}''(v'',v'))]$

"w is the only maximal node"

$\wedge \forall z(\mathrm{V}''(z) \wedge (z \neq v) \to \exists v'(\mathrm{E}''(v',z) \wedge \forall v''(\mathrm{V}''(v'') \wedge \mathrm{E}''(v'',z) \to v'' = v'))$

"all nodes except *v* have id=1"

$\wedge \forall z(\mathrm{V}''(z) \wedge (z \neq w) \to \exists v'(\mathrm{E}''(z,v') \wedge \forall v''(\mathrm{V}''(v'') \wedge \mathrm{E}''(z,v'') \to v'' = v')))$

"all nodes except *w* have od=1"

$\equiv$ "(V",E") is a linear subgraph of the graph (V,E), with minimal node *v* and maximal node *w*"

4. Path in 3rd order graph: $PATH_E^{3,2}(X,Y)$, where E is a 3rd order relation of arity 2, and X, Y are 2nd order relations of arity 1, meaning that there is a path in E from the set X to the set Y.

5. Linear graph: $LINEAR(V,E)$ (we will also use $LINEAR^{2,4}(V,E)$, where V is a 2nd order relation of arity 2, and E is a 2nd order relation of arity 4).

$\equiv\ [\forall v'''\, w'''((a)\vee(b))]\wedge(c)$

Formula (a) says " $(PATH_E(v''', w'''))$"

Formula (b) says " $(PATH_E(w''', v'''))$"

Formula (c) says " $\exists_{v\ w}$(v is the only minimal in (V,E), w is the only maximal in (V,E), every node except v has id=1 and every node except w has od=1)"

(a) $(\exists V''E''(\forall v', v''(E''(v', v'')\rightarrow V''(v')\wedge V''(v'')\wedge E(v', v''))$
$\wedge\forall v'(V''(v')\rightarrow V(v'))\wedge(V''(v''')\wedge V''(w'''))$
$\wedge\neg\exists v'(E''(v', v''')\wedge\forall v''[V''(v'')\wedge(v''\neq v''')\rightarrow\exists v'(E''(v', v''))]$
$\wedge\neg\exists v'(E''(w''', v')\wedge\forall v''[V''(v'')\wedge(v''\neq w''')\rightarrow\exists v'(E''(v'', v'))]$
$\wedge\forall z(V''(z)\wedge(z\neq v''')\rightarrow\exists v'(E''(v', z)\wedge\forall v''(V''(v'')\wedge E''(v'', z)\rightarrow v''=v'))$
$\wedge\forall z(V''(z)\wedge(z\neq w''')\rightarrow\exists v'(E''(z, v')\wedge\forall v''(V''(v'')\wedge E''(z, v'')\rightarrow v''=v')))$

(b) $\vee\exists V''E''(\forall v', v''(E''(v', v'')\rightarrow V''(v')\wedge V''(v'')\wedge E(v', v''))$
$\wedge\forall v'(V''(v')\rightarrow V(v'))\wedge(V''(v''')\wedge V''(w'''))$
$\wedge\neg\exists v'(E''(v', w''')\wedge\forall v''[V''(v'')\wedge(v''\neq w''')\rightarrow\exists v'(E''(v', v''))]$
$\wedge\neg\exists v'(E''(v''', v')\wedge\forall v''[V''(v'')\wedge(v''\neq v''')\rightarrow\exists v'(E''(v'', v'))]$
$\wedge\forall z(V''(z)\wedge(z\neq w''')\rightarrow\exists v'(E''(v', z)\wedge\forall v''(V''(v'')\wedge E''(v'', z)\rightarrow v''=v'))$
$\wedge\forall z(V''(z)\wedge(z\neq v''')\rightarrow\exists v'(E''(z, v')\wedge\forall v''(V''(v'')\wedge E''(z, v'')\rightarrow v''=v')))$

(c) $\wedge((V(v)\wedge V(w)$
$\wedge\neg\exists v'(E(v', v)\wedge\forall v''[V(v'')\wedge(v''\neq v)\rightarrow\exists v'(E(v', v''))]$
$\wedge\neg\exists v'(E(w, v')\wedge\forall v''[V(v'')\wedge(v''\neq w)\rightarrow\exists v'(E(v'', v'))]$
$\wedge\forall z(V(z)\wedge(z\neq v)\rightarrow\exists v'(E(v', z)\wedge\forall v''(V(v'')\wedge E(v'', z)\rightarrow v''=v'))$
$\wedge\forall z(V(z)\wedge(z\neq w)\rightarrow\exists v'(E(z, v')\wedge\forall v''(V(v'')\wedge E(z, v'')\rightarrow v''=v'))))$

6. $x=\text{"0"}\ \equiv\ \neg\exists y(y\neq x\wedge y\leq x)$

7. $x=\text{"1"}\ \equiv\ \exists y(y\neq x\wedge y\leq x)\wedge\forall z(z\leq x\rightarrow(z=x\vee z=y))$

4 Quantified Boolean Formulas

4.1 Definitions

1. Quantified Boolean Formulas: [GJ03, p107]

A quantified Boolean formula (QBF) is a formula of the form $F = Q_1 x_1 Q_2 x_2 ... Q_n x_n (E)$, where E is a Boolean expression involving the variables $x_1, x_2, ..., x_n$ and each Q_i is either "$\exists$" or "$\forall$".

QBF_k : Let X be an ordered set of propositional variables, let $X_1, X_2, X_3, ..., X_k$ be disjoint sub-sequences forming a partition of X, and let $X_i = \{x_{i1}, x_{i2}, ..., x_{il_i}\}$. A quantified Boolean formula over X is in QBF_k if it is of the form $\varphi_k = \exists X_1 \forall X_2 \exists X_3 ... Q X_k (\varphi'(x_{11}...x_{1l_1}, x_{21}...x_{2l_2}, x_{31}...x_{3l_3}, ..., x_{k1}...x_{kl_k}))$, where the quantifier Q is $\exists$ if k is odd and $\forall$ if k is even. φ' is a quantifier-free Boolean formula over the variables in X. X_i denotes a sequence of different variables from the set X, and $\exists X_i$ denotes $\exists x_{i1} \exists x_{i2} \exists x_{i3} ... \exists x_{il_i}$ $\forall X_i$ denotes $\forall x_{i1} \forall x_{i2} \forall x_{i3} ... \forall x_{il_i}$ etc.

2. An alternating valuation is a finite structure $a_v = (V, E, B)$ such that:

1) (V, E) is a directed graph, which is an out-tree that has all the leaves at the same depth d, for some natural d.

2) The output degree of any non leaf node is ≤ 2.

3) For every $1 \leq k \leq d$, all the nodes at depth k have the same output degree.

4) $B:V\to\{0,1\}$ is a total function such that for every pair of nodes a, b which are siblings, it is $B(a)\neq B(b)$.

3. Let φ be a QBF_k formula, for some natural $k>0$, let $l_1,...,l_k>0$, respectively, be the lengths of the alternating quantifier blocks of φ, and let $a_v=(V,E,B)$ be an alternating valuation. We say that a_v is an alternating valuation suitable for φ, if the following holds:

a) The depth of the tree (V,E) is $d=l_1+...+l_k-1$.

b) For every $1\le i\le l_1+...+l_k$:

i) if $1\le i\le l_1$ or $l_1+l_2+1\le i\le l_1+l_2+l_3$, … or $l_1+l_2+...+l_{k'-1}+1\le i\le l_1+l_2+...+l_{k'}$, then all the nodes at depth $i-1$ in (V,E) have no siblings, where $1\le k'\le k$ is the index of the last existential quantifier block in φ;

ii) if $l_1+1\le i\le l_1+l_2$, or $l_1+l_2+l_3+1\le i\le l_1+l_2+l_3+l_4$, … or $l_1+l_2+...+l_{k''-1}+1\le i\le l_1+l_2+...+l_{k''}$, then all the nodes at depth $i-1$ in (V,E) have exactly one sibling, where $2\le k''\le k$ is the index of the last universal quantifier block in φ. Note that there might not be such k'' in φ.

4. Let φ be a QBF_k formula, for some natural $k > 0$, and let $a_v = (V, E, B)$ be an alternating valuation that is suitable for φ. A leaf valuation in a_v is a structure $l_v = (V', E', B')$ such that (V', E') is a path in (V, E) from the root to one leaf, and $B' = B|_{V'}$, i.e., $B' = \{(v,b) : v \in V' \wedge (v,b) \in B\}$.

5. A leaf valuation l_v in a_v corresponds to a valuation v for φ, where for $0 \le i \le d$, if the node a has depth i in (V', E'), then $v(x_{jh}) = B(a)$, where $0 \le j \le k$ is the greatest natural such that $i+1 > l_1 + l_2 + ... + l_{j-1}$ and $h = i+1-(l_1 + l_2 + ... + l_{j-1})$.

   That is, every depth $0 \le i \le d$ in a_v corresponds to one variable in φ, namely the variable that is quantified in the $(i+1)-th$ place in the quantifier prefix of φ. Correspondingly, every node in l_v corresponds to one variable in φ, i.e., the $i-th$ node in (V', E') corresponds to the variable that is quantified in the $(i+1)-th$ place in the quantifier prefix of φ.

6. Let $\varphi \in QBF_k$, for some $k \ge 1$, and let a_v be an alternating valuation suitable for φ. We say that a_v satisfies φ if for every leaf valuation l_v in a_v, it is the case that l_v satisfies φ, i.e., the valuation v that corresponds to l_v according to the definition above, satisfies φ; in symbols: $v \models \varphi$.

7. Let $k \ge 1$. $SATQBF_k$ is the set of formulas $\varphi \in QBF_k$, such that there is an alternating valuation suitable for φ that satisfies φ.

8. $SATQBF$ is the set of formulas $\varphi \in QBF_k$, for some $k \geq 1$, such that there is an alternating valuation suitable for φ that satisfies φ.

9. Note that with the definition in 3, we are building a tree to represent the assignment of values to variables where each variable in an existentially quantified block has only one value assigned to it, and each variable in a universally quantified block gives rise to two paths in the tree and both of these paths must be able to be given a valuation such that the quantifier free part of φ is satisfied, in order for the alternating valuation to satisfy φ. (See ***Figure 8***)

5 Expressing $SATQBF$ and $SATQBF_k$ in Logic

5.1 Complexity of the problems

We briefly show in this sub section that there are formulas in SO and TO that express the queries $SATQBF_k$ and $SATQBF$, respectively. Note, however, that to the best of our knowledge these formulas are not known.

5.1.1 $SATQBF_k$ can be expressed in Second-Order Logic

It is known that, for $k \geq 1$, $SATQBF_k$ is complete for Σ_k^P, where Σ_k^P is defined as follows. ([Sto76])

Definition The polynomial-time hierarchy (PH)

is $\{\Sigma_k^P, \Pi_k^P, \Delta_k^P : k \geq 0\}$, where $\Sigma_0^P = \Pi_0^P = \Delta_0^P = P$; and for $k \geq 0$,

$$\Sigma_{k+1}^P = NP(\Sigma_k^P) = NP^{NP^{NP^{\cdot^{\cdot^{NP}}}}} \quad \text{(There is a stack of } k+1 \text{ NP's)}$$

$$\Pi_{k+1}^P = co - NP(\Sigma_k^P),$$

$$\Delta_{k+1}^P = P(\Sigma_k^P).$$

$$\text{We also define } PH = \bigcup_{k=0}^{\infty} \Sigma_k^P.$$

Where, if *C*, *D*, are Turing Machine complexity classes, C^D denotes the class of queries computed by machines in *C* which have an oracle in the class *D*.

In particular, note that $\Sigma_1^P = NP$ and $\Pi_1^P = co - NP$. PH possesses the following inclusion structure:

$$\Sigma_k^P \bigcup \Pi_k^P \subseteq \Delta_{k+1}^P \subseteq \Sigma_{k+1}^P \bigcap \Pi_{k+1}^P \text{ for all } k \geq 0.$$

Corollary 7.22 ([Imm99]) A Boolean query is in PH, iff it is second-order expressible, that is PH=SO.

Hence, we know that, for every $k \geq 1$ $SATQBF_k$ can be expressed in SO. In fact, $SATQBF_k \in \sum_k^1$.

5.1.2 SATQBF can be expressed in Third-Order Logic

SATQBF $\in \exists$TO and hence $\in$TO, because of the following 3 facts:

1. $PSPACE \subseteq DTIME(2^{n^{O(1)}}) \subseteq NTIME(2^{n^{O(1)}})$ [GJ03]

2. **Theorem 4.4** ([Sto76], p19): SATQBF is PSPACE-complete.

3. $\exists TO = NTIME(2^{n^{O(1)}})$ and $TO = \bigcup_k NTIME(2^{n^{O(1)}})^{NP^{NP^{\cdot^{\cdot^{\cdot^{NP}}}}}}$ (there is a stack of k $NP's$). [HT06]

Hence, we know that $SATQBF$ can be expressed in TO.

5.2 Expressing $SATQBF_k$ in Second-Order Logic

In this subsection we present a detailed construction of an SO formula which expresses $SATQBF_k$.

To encode the input formula, we use the following vocabulary (following [MP96] and [Fer08]):

$$\sigma = <\leq, P_\neg, P_\vee, P_\wedge, P_\exists, P_\forall, P_(, P_), P_X, P_| >$$

Then: $I = < D^I, \leq^I, P_\neg{}^I, P_\vee{}^I, P_\wedge{}^I, P_\exists{}^I, P_\forall{}^I, P_(^I, P_)^I, P_X{}^I, P_|^I >$ represents a QBF_k formula φ.

We assume that the input formula φ, except for the quantifier prefix Q, is fully parenthesized, there is at least one pair of matching parentheses, and every sub formula in φ' which involves a unary or binary operator is enclosed in parentheses. We further assume that there is at least one quantifier in φ, and that there are no parentheses in the quantifier prefix.

$$\varphi = \exists x_{11}...\exists x_{1l_1} \forall x_{21}...\forall x_{2l_2} \exists x_{31}...\exists x_{3l_3} .....Q...x_{kl_k} (\varphi'(x_{11}...x_{1l_1}, x_{21}...x_{2l_2}, x_{31}...x_{3l_3}, ..., x_{k1}...x_{kl_k}))$$

Example: let $\varphi = \exists x_1 \exists x_2 \forall x_3 ((\neg x_1 \vee x_3) \wedge (\neg x_2 \vee x_3))$

As a linear graph, which is defined as the successor relation induced by $\leq^I$:

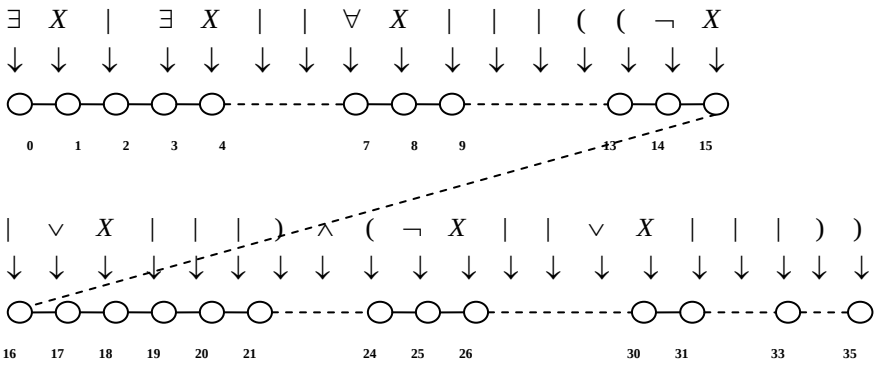


Figure 6

The σ - structure that encodes φ is as follows:

$$I = < D^I, \leq^I, P_\neg{}^I, P_\vee{}^I, P_\wedge{}^I, P_\exists{}^I, P_\forall{}^I, P_(^I, P_)^I, P_X{}^I, P_|^I >$$

where

$D^I = \{0,1,...,35\}$

$\leq^I$: total order in D^I

$P_\neg{}^I = \{14, 25\}$

$P_\vee{}^I = \{17, 29\}$

$P_\wedge{}^I = \{23\}$

$P_\exists{}^I = \{0, 3\}$

$P_\forall{}^I = \{7\}$

$P_(^I = \{12, 13, 24\}$

$P_)^I = \{22, 34, 35\}$

$P_X{}^I = \{1, 4, 8, 15, 18, 26, 30\}$

$P_|^I = \{2, 5, 6, 9, 10, 11, 16, 19, 20, 21, 27, 28, 31, 32, 33\}$

5.2.1 Informal Expression

Let $k \geq 1$. We can build an SO formula to express the problem $SATQBF_k$ in four levels of abstraction.

1st Level:

"$\exists$ alternating valuation a_v suitable for φ which satisfies φ". This can be expressed as "$\exists$ a partial valuation on block 1, such that $\forall$ partial valuation on block 2, $\exists$ a partial valuation on block 3, …" such that all of these valuations satisfy the quantifier free part φ' of φ. This is clearly equivalent to the following:

2nd Level:

(2.1) " $\exists$ Partial valuation v_1 on $\{x_{11},...,x_{1l_1}\}$, $\forall$ partial valuation v_2 on $\{x_{21},...,x_{2l_2}\}$,…, $\exists/\forall$ partial valuation v_k on $\{x_{k1},...,x_{kl_k}\}$ such that …"

(2.2) "… the valuation $v = v_1 \cup v_2 \cup ... \cup v_k$ satisfies φ'" Recall that φ' is the quantifier free part of φ.

Following our definitions in 4.1 above, we can think of an alternating valuation as a binary tree, as in ***Figure 8***. In that tree, that represents an alternating valuation suitable for φ, each leaf node determines a unique path from the root to that node. We represent that path with the linear graph G_t which together with the function B_t represents what we have defined as a

leaf valuation. Note that the alternating valuation satisfies the input formula φ, if and only if every leaf valuation satisfies φ'.

We will use linear graphs $G_i = (V_i, E_i)$ with functions $B_i : V_i \to \{0,1\}$ to represent the partial valuations v_i (See ***Figure 7***). Correspondingly, we will use a linear graph $G_t = (V_t, E_t)$ with a function $B_t : V_t \to \{0,1\}$ to represent the leaf valuation v. So, each leaf valuation is represented by a corresponding pair $< G_t, B_t >$.

3rd Level:

We give next a more detailed expression of the two abstract expressions of level 2, labelled **(2.1), (2.2)**.

5.2.1.1 Expressing (2.1): Encoding each leaf valuation for the input formula φ, using the graph G_t

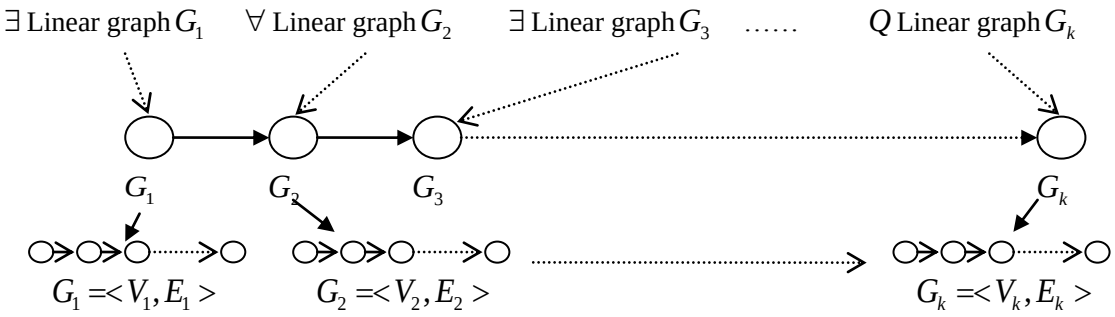


Figure 7

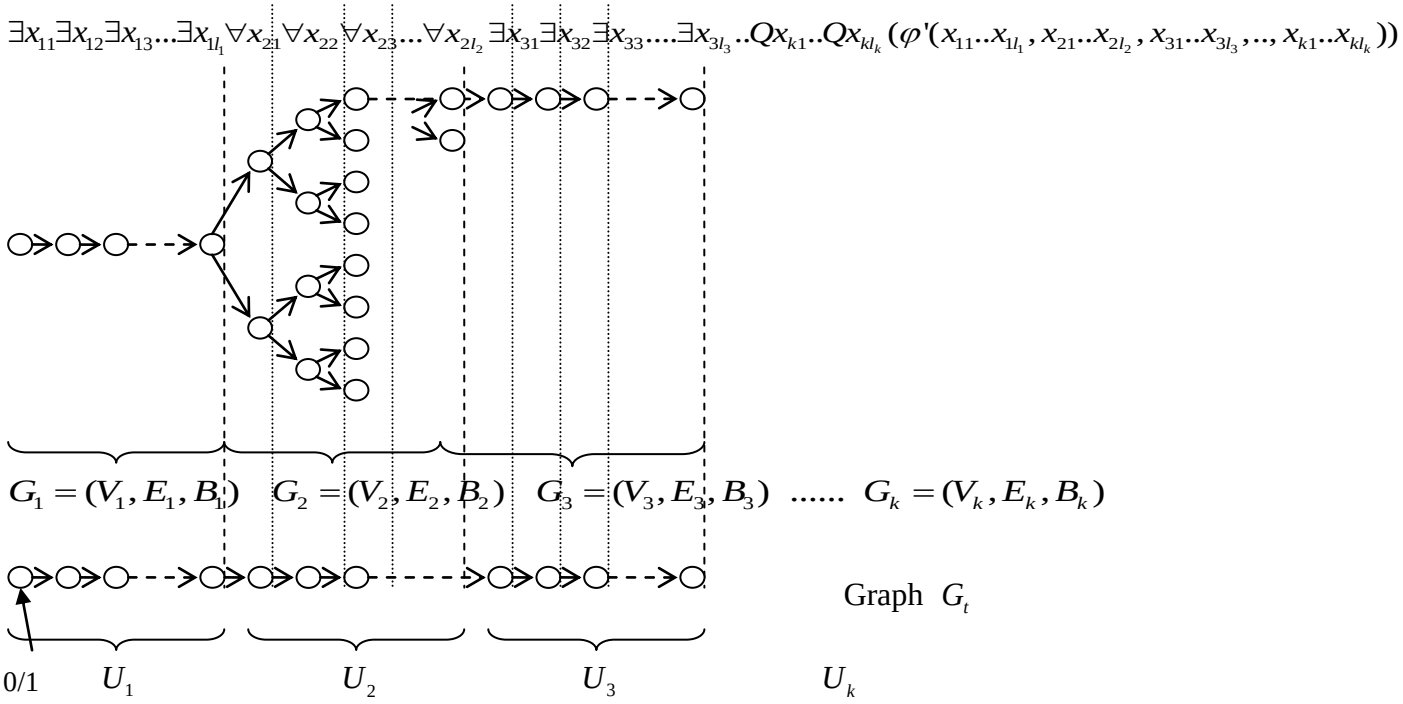


Figure 8

Let $X_1, X_2, ..., X_k$ be disjoint sub-sequences of propositional variables.

If we consider the quantifier prefix of the input formula $\exists X_1 \forall X_2 \exists X_3 ... QX_k$ as a tree, then the depth of the tree is $\sum_{i=1}^{k} |V_i| = |V_1| + |V_2| + ... + |V_k|$. And if x_i in the quantifier-free formula φ' is quantified in order j-th in φ, then its value must be B (j-th node in G_t).

Expression **(2.1)** can then be expressed as follows:

$\equiv \exists$set V_1 and binary relations E_1 and $B_1, \forall$set V_2 and binary relations E_2 and B_2,
$\exists ..., Q_k$ set V_k and binary relations E_k and $B_k, (\exists V_t^1, E_t^2, B_t^2$

$([$"$G_t = < V_t^1, E_t^2 >$ is a linear graph"] **(1)**

$\wedge$**[**"The length of G_t is equal to the number of variables
in the quantifier prefix of φ"] **(2)**

$\wedge$["$G_1 = < V_1, E_1 >, G_3 = < V_3, E_3 >, ..., G_{k'} = < V_{k'}, E_{k'} >$ are linear graphs"] **(3)**
(where k' is the index of the last $\exists$ quantifier in the prefix)

$\wedge$["$B_1 : V_1 \to \{0,1\}, B_3 : V_3 \to \{0,1\}, ..., B_{k'} : V_{k'} \to \{0,1\}$ are total functions"] **(4)**

$\wedge$["the lengths of the linear graphs $G_1, G_3, ..., G_{k'}$ are equal to the lengths of
the corresponding quantifier blocks in φ"] **(5)**

$\wedge [_1 [_2$("$V_1, V_2, V_3, ..., V_k$ are disjoint sets" **(6)**

$\wedge$["$G_2 = < V_2, E_2 >, G_4 = < V_4, E_4 > ... G_{k"} = < V_{k"}, E_{k"} >$ are linear graphs"] **(7)**
(where $k"$ is the index of the last $\forall$ quantifier in the prefix)

$\wedge$["$B_2 : V_2 \to \{0,1\}, B_4 : V_4 \to \{0,1\}, ..., B_{k"} : V_{k"} \to \{0,1\}$ are total functions"] **(8)**

$\wedge$["the lengths of the linear graphs $G_2, G_4, ..., G_{k"}$ are equal to the lengths of
the corresponding quantifier blocks in φ"**]**) **(9)**

$\wedge \exists U_1^2, U_2^2 ... U_k^2$("$U_1$ is a partial bijection: $G_1 \to$1st part of G_t"

$\wedge$"U_2 is a partial bijection: $G_2 \to$ 2nd part of G_t"

$\wedge ... ... \wedge$"U_k is a partial bijection: $G_k \to$ k_th part of G_t") **(10)**

$\wedge$"$B_t : V_t \to \{0,1\}$ is a total function that *coincides* with $B_1, B_2, ..., B_k$"
$\equiv$"$B_t = B_1 \bigcup B_2 \bigcup ... \bigcup B_k$"$]_2 \to$ **(11)**

(Level 4)

(1) "The graph G_t is a linear graph": $LINEAR(V_t, E_t)$

(2) "The length of G_t is equal to the number of variables in the quantifier prefix of φ"

This statement is implied by the following statement:

"V_p is a partial surjective injection from the quantifier prefix of φ to G_t, which maps every X to its corresponding node in G_t, and which preserves $\leq^I$ and E_t."

This statement is expressed in further detail in (1) in 5.2.1.2.1

(3) "$G_1 =< V_1, E_1 >, G_3 =< V_3, E_3 >, ..., G_{k'} =< V_{k'}, E_{k'} >$ are linear graphs":
(where k' is the index of the last $\exists$ quantifier in the prefix)

$$LINEAR(V_1, E_1) \wedge LINEAR(V_3, E_3) \wedge ... \wedge LINEAR(V_{k'}, E_{k'})$$

(4) "$B_1 : V_1 \to \{0,1\} \wedge B_3 : V_3 \to \{0,1\} \wedge ... \wedge B_{k'} : V_{k'} \to \{0,1\}$ are total functions"

$$\begin{aligned}
&\wedge [\forall t, p, p' [B_1(t, p) \wedge B_1(t, p') \to (p = p') && \text{"function"} \\
&\qquad \wedge (V_1(t) \to \exists p(B_1(t, p))) && \text{"total"} \\
&\qquad \wedge (B_1(t, p) \to (p = \text{"0"} \vee p = \text{"1"}))] \\
&\wedge \forall t, p, p' [B_3(t, p) \wedge B_3(t, p') \to (p = p') && \text{"function"} \\
&\qquad \wedge (V_3(t) \to \exists p(B_3(t, p))) && \text{"total"} \\
&\qquad \wedge (B_3(t, p) \to (p = \text{"0"} \vee p = \text{"1"}))] \\
&\wedge ..... \\
&\wedge \forall t, p, p' [B_{k'}(t, p) \wedge B_{k'}(t, p') \to (p = p') && \text{"function"} \\
&\qquad \wedge (V_{k'}(t) \to \exists p(B_{k'}(t, p))) && \text{"total"} \\
&\qquad \wedge (B_{k'}(t, p) \to (p = \text{"0"} \vee p = \text{"1"}))]]
\end{aligned}$$

(5) If $k' \neq k,$ then (by $\alpha_{k'1}$ we mean the formula template α_{i1} instantiated with $i = k'$):

$$\bigwedge_{i=1,3,5,...,k'} \exists L'^{2} v_1 v_2 v_3 ... v_{k'} v_{k'+1} (\alpha_{k'1} \wedge \alpha_{i2})$$

If $k' = k,$ then (by $\alpha_{k'-2,1}$ we mean the formula template α_{i1} instantiated with $i = k'-2$):
($k'-2$ is the previous to the last $\exists$ block, and the subformulas $\beta_1, \beta_2, \beta_3$
take care of the last quantifier block).

$$\left(\bigwedge_{i=1,3,5,...,k'-2} \exists L'^{2} v_1 v_2 v_3 ... v_{k'-1} (\alpha_{k'-2,1} \wedge \alpha_{i2}) \right) \wedge \exists L'^{2} v_1 v_2 v_3 ... v_k v_e (\beta_1 \wedge \beta_2 \wedge \beta_3)$$

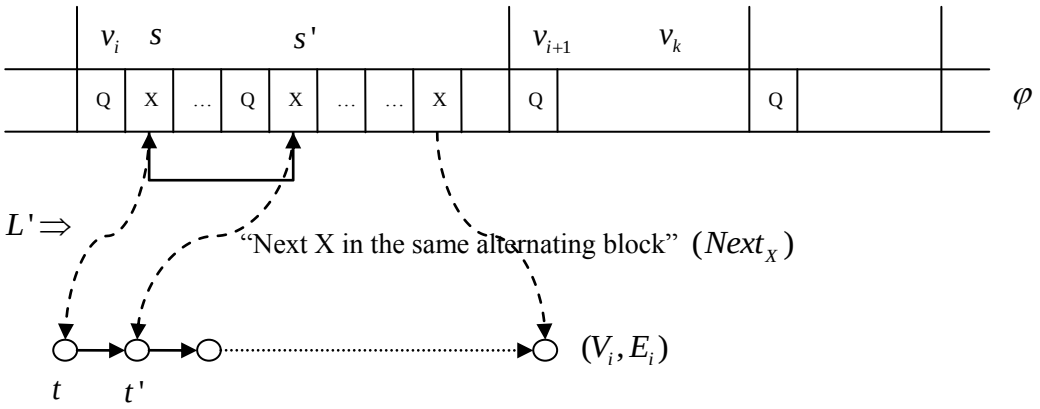


Figure 9

"for $1 \leq j \leq i,$ v_j is the first quantifier of the j-th alternating quantifier block:"
(when i is not the index of the last block)

$$\alpha_{i1} \equiv ({}_0 P_{\exists}(v_1) \wedge P_{\forall}(v_2) \wedge P_{\exists}(v_3) \wedge ... \wedge P_Q(v_{i+1}) \wedge v_1 = "0"$$

"where P_Q is $P_{\exists}$ if $(i+1)$ is odd, or $P_{\forall}$ if $(i+1)$ is even"

$$\begin{aligned}
&\wedge [PATH_{\leq}(v_1, v_2) \wedge PATH_{\leq}(v_2, v_3) \wedge ... \wedge PATH_{\leq}(v_i, v_{i+1})] \\
&\wedge [\neg \exists v' (PATH_{\leq}(v_1, v') \wedge PATH_{\leq}(v', v_2) \wedge v' \neq v_1 \wedge v' \neq v_2 \wedge P_{\forall}(v'))] \\
&\wedge [\neg \exists v' (PATH_{\leq}(v_2, v') \wedge PATH_{\leq}(v', v_3) \wedge v' \neq v_2 \wedge v' \neq v_3 \wedge P_{\exists}(v'))] \\
&\wedge ... \\
&\wedge [\neg \exists v' (PATH_{\leq}(v_i, v') \wedge PATH_{\leq}(v', v_{i+1}) \wedge v' \neq v_i \wedge v' \neq v_{i+1} \wedge P_{Q'}(v'))])_0
\end{aligned}$$

"where $P_{Q'}$ is $P_{\forall}$ if i is odd, or $P_{\exists}$ if i is even"

"for $1 \le j \le i,\ v_j$ is the first quantifier of the j-th alternating quantifier block:"
("for $i = k,\ v_e$ is the last element in the block $i,$ which must be |")

$$\beta_1 \equiv ({}_0 P_{\exists}(v_1) \wedge P_{\forall}(v_2) \wedge P_{\exists}(v_3) \wedge ... \wedge P_Q(v_k) \wedge P_{|}(v_e) \wedge v_1 = \text{"0"}$$

"where P_Q is $P_{\exists}$ if k is odd, or $P_{\forall}$ if k is even"

$$\wedge [PATH_{\le}(v_1, v_2) \wedge PATH_{\le}(v_2, v_3) \wedge ... \wedge PATH_{\le}(v_k, v_e)]$$
$$\wedge [\neg \exists v'(PATH_{\le}(v_1, v') \wedge PATH_{\le}(v', v_2) \wedge v' \neq v_1 \wedge v' \neq v_2 \wedge P_{\forall}(v'))]$$
$$\wedge [\neg \exists v'(PATH_{\le}(v_2, v') \wedge PATH_{\le}(v', v_3) \wedge v' \neq v_2 \wedge v' \neq v_3 \wedge P_{\exists}(v'))]$$
$$\wedge ...$$
$$\wedge [\neg \exists v'(PATH_{\le}(v_k, v') \wedge PATH_{\le}(v', v_e) \wedge v' \neq v_k \wedge v' \neq v_e \wedge P_{Q'}(v'))])_0$$

"where $P_{Q'}$ is $P_{\forall}$ if k is odd, or $P_{\exists}$ if k is even"

"L' is a bijection from the X in the i-th alternating quantifier block to V_i,
which preserves $Next_X$ and E_i:"
(when i is not the index of the last block)

$$\alpha_{i2} \equiv ({}_0 \forall v'[(PATH_{\le}(v_i, v') \wedge PATH_{\le}(v', v_{i+1}) \wedge v' \neq v_{i+1} \wedge P_X(v')) \leftrightarrow \exists y'(L'(v', y'))]$$
"domain of L' "

$$\wedge \forall y'(V_i(y') \rightarrow \exists v'(L'(v', y'))) \quad \text{"}L'\text{ is surjective"}$$

$$\wedge \forall s, t, s', t'[[(L'(s,t) \wedge L'(s',t') \wedge s \neq s' \wedge PATH_{\le}(v_i, s) \wedge PATH_{\le}(s', v_{i+1}) \wedge PATH_{\le}(s, s')$$
$$\wedge (\neg \exists z'(PATH_{\le}(s, z') \wedge PATH_{\le}(z', s') \wedge z' \neq s \wedge z' \neq s' \wedge P_X(z')))) \rightarrow E_i(t, t')]$$
"L' preseves $Next_X$ in the *ith* alternating block and E_i (implies injectivity)"

$$\wedge \forall v' y'[L'(v', y') \rightarrow V_i(y')]$$
$$\wedge \forall v' y' z'([L'(v', y') \wedge L'(v', z')] \rightarrow (y' = z'))])_0 \quad \text{"}L'\text{ is a function"}$$

"L' is a bijection from the X in the i-th alternating quantifier block to V_i,
which preserves $Next_X$ and E_i:"
(when $i = k$)

$\beta_2 \equiv ({}_0\forall v'[(PATH_{\leq}(v_k, v') \wedge PATH_{\leq}(v', v_e) \wedge P_X(v')) \leftrightarrow \exists y'(L'(v', y'))]$
"domain of L' "

$\wedge \forall y'(V_k(y') \rightarrow \exists v'(L'(v', y')))$ "L' is surjective"

$\wedge \forall s, t, s', t'[[(L'(s,t) \wedge L'(s',t') \wedge s \neq s' \wedge PATH_{\leq}(v_k, s) \wedge PATH_{\leq}(s', v_e) \wedge PATH_{\leq}(s, s')$
$\wedge (\neg\exists z'(PATH_{\leq}(s, z') \wedge PATH_{\leq}(z', s') \wedge z' \neq s \wedge z' \neq s' \wedge P_X(z')))) \rightarrow E_k(t, t')]$
"L' preseves $Next_X$ in the k-*th* alternating block and E_k (implies injectivity)"

$\wedge \forall v' y'[L'(v', y') \rightarrow V_k(y')]$
$\wedge \forall v' y' z'([L'(v', y') \wedge L'(v', z')] \rightarrow (y' = z'))]_0$ "L' is a function"

"For $i = k$, v_e is the last | for the last quantifier in the quantifier prefix":

$\beta_3 \equiv ({}_0\forall v'((SUC_{\leq}(v_e, v') \wedge \neg P_{|}(v')) \wedge P_{|}(v_e)$
$\wedge \forall v'(PATH_{\leq}(v_e, v') \rightarrow (\neg P_{\exists}(v') \wedge \neg P_{\forall}(v')))$ "there are no quantifiers after v_e"

$\wedge \exists z' y' w' \forall v'[PATH_{\leq}(y', v_e) \wedge ([PATH_{\leq}(v', v_e) \wedge PATH_{\leq}(y', v')] \rightarrow P_{|}(v'))$
"w' is the last quantifier in the block k"
"z' is the X of the variable quantified"
"y' is the first | for that variable"

$\wedge P_Q(w') \wedge P_X(z') \wedge SUC_{\leq}(z', y') \wedge SUC_{\leq}(w', z')])_0$
"where P_Q is $P_{\exists}$ if k is odd, or P_Q is $P_{\forall}$ if k is even"

(6) $\wedge$ "$[(V_1 \bigcap V_2 = \phi) \wedge (V_1 \bigcap V_3 = \phi) \wedge ... \wedge (V_1 \bigcap V_k = \phi)$
$\wedge (V_2 \bigcap V_3 = \phi) \wedge (V_2 \bigcap V_4 = \phi) \wedge ... \wedge (V_2 \bigcap V_k = \phi) \wedge ... \wedge (V_{k-1} \bigcap V_k = \phi)]$"

where "$V_i \bigcap V_j = \phi$" is $\forall x((V_i(x) \rightarrow \neg V_j(x)) \wedge (V_j(x) \rightarrow \neg V_i(x)))$

(7) "$G_2 = < V_2, E_2 >, G_4 = < V_4, E_4 > ... G_{k"} = < V_{k"}, E_{k"} >$ are linear graphs":

(where $k"$ is the index of the last $\forall$ quantifier in the prefix)

$LINEAR(V_2, E_2) \wedge LINEAR(V_4, E_4) \wedge ... \wedge LINEAR(V_{k"}, E_{k"})$

(8) "$B_2 : V_2 \to \{0,1\} \wedge B_4 : V_4 \to \{0,1\} \wedge \ldots \wedge B_{k"} : V_{k"} \to \{0,1\}$ are total functions"

$\wedge [\forall t, p, p' [B_2(t, p) \wedge B_2(t, p') \to (p = p')$ "function"

$\wedge (V_2(t) \to \exists p(B_2(t, p)))$ "total"

$\wedge (B_2(t, p) \to (p = "0" \vee p = "1"))]$

$\wedge \forall t, p, p' [B_4(t, p) \wedge B_4(t, p') \to (p = p')$ "function"

$\wedge (V_4(t) \to \exists p(B_4(t, p)))$ "total"

$\wedge (B_4(t, p) \to (p = "0" \vee p = "1"))]$

$\wedge \ldots\ldots$

$\wedge \forall t, p, p' [B_{k"}(t, p) \wedge B_{k"}(t, p') \to (p = p')$ "function"

$\wedge (V_{k"}(t) \to \exists p(B_{k"}(t, p)))$ "total"

$\wedge (B_{k"}(t, p) \to (p = "0" \vee p = "1"))]]$

(9) If $k" \neq k,$ then (by $\alpha_{k"1}$ we mean the formula template α_{i1} instantiated with $i = k"$):

$$\bigwedge_{i=2,4,6,\ldots,k"} \exists L^{'2} v_1 v_2 v_3 \ldots v_{k"} v_{k"+1} (\alpha_{k"1} \wedge \alpha_{i2})$$

If $k" = k,$ then (by $\alpha_{k"-2,1}$ we mean the formula template α_{i1} instantiated with $i = k" - 2$):
($k" - 2$ is the previous to the last $\forall$ block, and the subformulas $\beta_1, \beta_2, \beta_3$
take care of the last quantifier block)

$$\left(\bigwedge_{i=2,4,6,\ldots,k"-2} \exists L^{'2} v_1 v_2 v_3 \ldots v_{k"-1} (\alpha_{k"-2,1} \wedge \alpha_{i2}) \right) \wedge \exists L^{'2} v_1 v_2 v_3 \ldots v_k v_e (\beta_1 \wedge \beta_2 \wedge \beta_3)$$

where the subformulas $\alpha_{i1}, \alpha_{i2}, \beta_1, \beta_2$ and β_3 are the same as in (5) above.

(10) $\wedge \exists \mathbf{U}_1^2 \mathbf{U}_2^2 \ldots \mathbf{U}_k^2$ ("

$\mathbf{U}_1 : V_1 \to V_t$ – Total injection,

– preserves E_1, E_t,

– includes first node in V_t as U_1(first node in E_1),

$\mathbf{U}_2 : V_2 \to V_t$ – Total injection,

– preserves E_2, E_t,

– includes successor in E_t of node U_1(last node in E_1) in V_t as U_2(first node in E_2),

$\mathbf{U}_3 : V_3 \to V_t$ – Total injection,

– preserves E_3, E_t,

– includes successor in E_t of node U_2(last node in E_2) in V_t as U_3(first node in E_3),

..., and

$\mathbf{U}_k : V_k \to V_t$ – Total injection,

– preserves E_k, E_t,

– includes successor in E_t of node U_{k-1}(last node in E_{k-1}) in V_t as U_k(first node in E_k)")

$$\equiv \wedge \exists U_1^2 U_2^2 ... U_k^2 (...$$

$$\wedge \forall x, y, t, u[((U_k(x,y) \wedge U_k(x,t)) \rightarrow y = t) \quad \text{"function"}$$

$$\wedge (U_k(x,y) \wedge U_k(u,y)) \rightarrow x = u) \quad \text{"injection"}$$

$$\wedge (V_k(x) \rightarrow \exists y(U_k(x,y))) \quad \text{"total"}$$

$$\wedge (U_k(x,y) \rightarrow (V_k(x) \wedge V_t(y)) \quad \text{"}U_k : V_k \rightarrow V_t\text{"}$$

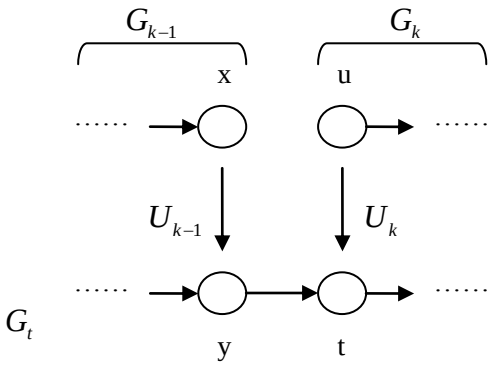
$$\wedge ((U_k(x,y) \wedge U_k(u,t) \wedge E_k(x,u)) \rightarrow E_t(y,t)) \quad \text{"preserves } E_k\text{"}$$

$$\wedge ((U_k(x,y) \wedge U_k(u,t) \wedge E_t(y,t)) \rightarrow E_k(x,u)) \quad \text{"preserves } E_t\text{"}$$

$$\wedge ((U_{k-1}(x,y) \wedge \neg \exists v(E_{k-1}(x,v)) \wedge E_t(y,t) \wedge \neg \exists v(E_k(v,u) \wedge V_k(u)))$$

$$\rightarrow U_k(u,t))])$$

"includes successor in E_t of node U_{k-1}(last node in E_{k-1})
in V_t as U_k(first node in E_k)"



(11) "B_t *coincides* with $B_1, B_2, ..., B_k$"

$$\forall xytpp'[(B_1(t,p) \wedge U_1(t,y) \wedge B_t(y,p')) \rightarrow p = p']$$

$$\wedge$$

$$\forall xytpp'[(B_2(t,p) \wedge U_2(t,y) \wedge B_t(y,p')) \rightarrow p = p']$$

$$\wedge$$

...

$$\wedge$$

$$\forall xytpp'[(B_k(t,p) \wedge U_k(t,y) \wedge B_t(y,p')) \rightarrow p = p']$$

5.2.1.2 Expressing (2.2)

(2.2) is equivalent to the following informal expression, which we will express in two parts:

$...\exists V_p^{\,2}, C^2, E_C^4, ST^1, E_{ST}^2, M^3, C_\wedge^2, C_\vee^2, C_\neg^2, C_(^2, C_)^2, C_1^2, C_0^2, H_\phi^3($

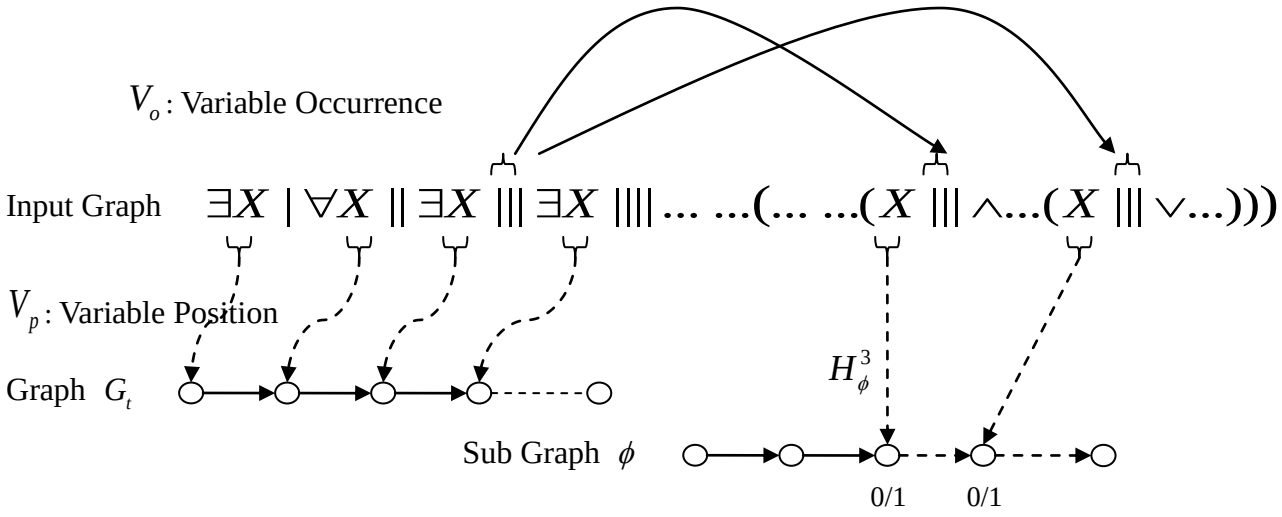
(2.2a) $[_2[$ "There is a formula ϕ on {0,1} which corresponds to the formula φ with the leaf valuation represented by $< G_t, B_t > \ldots$" $]\wedge$

(2.2b) ["...such that the formula ϕ is true"]$)]_2]_1)_0$

5.2.1.2.1 Expressing (2.2a):

Now we need to represent the relationships among the input graph, the graph G_t and the quantifier free part ϕ of the input formula:

$$\exists X \mid \forall X \mid\mid \exists X \mid\mid\mid ...QX \mid\mid\mid ... \mid (\varphi'(X \mid, X \mid\mid, X \mid\mid\mid, ..., X \mid\mid\mid ... \mid)$$



ϕ: A quantifier free formula on: $\{(,), \wedge, \vee, \neg, 1, 0\}$

For instance: $(((1 \wedge \neg 0) \vee (1 \wedge 0)) \wedge ...(...))$

Figure 10

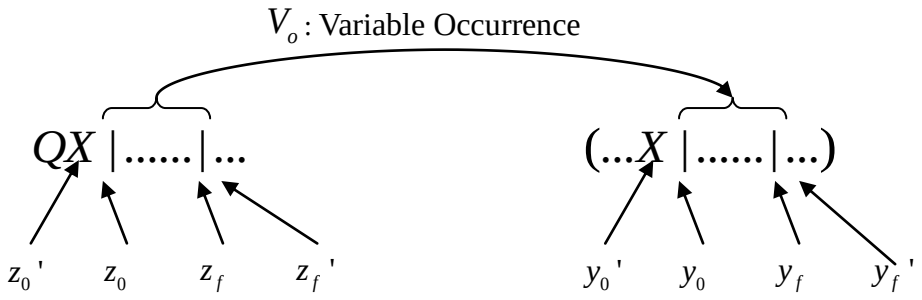


Figure 11

This expression is showed below based upon ***Figure 10***.

["V_p is a partial bijection from the quantifier prefix of φ (restricted to the X in the quantifier prefix) to V_t, which maps every X to its corresponding node in G_t, and which preserves $\leq^I$ and E_t."] **(1)**

$\wedge$["H_ϕ is a partial surjective injection from the quantifier free part of φ to the first formula in (C, E_C) (see Figures 12, 13), which maps every "X" in the quantifier free part to the corresponding position in the first formula in (C, E_C) which we denote by ϕ, which preserves $\wedge,\vee,(,),\neg$, and $\leq_I$ and E_C, and which ignores "|"."] **(2)**

$\wedge$["$\forall V_o$ that is a bijection from "|...|" in "$QX\,|...|$" to "|...|" in "$(...X\,|...|...)$", linking the quantification of a variable with each occurrence of the variable in the quantifier free part of φ, where z_0 is the root in $dom(V_o)$, z_f is the leaf in $dom(V_o)$, y_0 is the root in $ran(V_o)$, and y_f is the leaf in $ran(V_o)$. The variable in φ which corresponds to the function V_o is replaced in ϕ by the value assigned to that variable by the leaf valuation $<G_t, B_t>$. (See **Figures 10, 11)** Note that ϕ is represented in (C, E_C) starting in the node M(first node in (ST, E_{ST})) and ending in the node $E_C^{-1}(M$(second node in (ST, E_{ST}))) and is equivalent to the formula φ without quantifier prefix, with the variables replaced by 0 or 1 according to the leaf valuation in $<G_t, B_t>$."]
(More explanation is given in **5.2.1.2.2**. See **Figures 12 and 13**) **(3)**

(Level 4)

(1)$(\forall x,y,z(V_p(x,y) \wedge V_p(x,z) \rightarrow (y=z)$ "V_p is a function"

$\wedge (V_p(x,y) \wedge V_p(z,y) \rightarrow x=z)$ "V_p is injective"

$\wedge ((P_X(x) \wedge PRED_{\leq}(x,z) \wedge [P_{\exists}(z) \vee P_{\forall}(z)]) \leftrightarrow \exists y(V_t(y) \wedge V_p(x,y)))$

"domain of V_p","range of V_p"

"V_p is a partial bijection: X in quantifier prefix of $\varphi \rightarrow V_t$ in G_t"

$\wedge \forall s,t,s',t'([V_p(s,s') \wedge V_p(t,t') \wedge E_t(s',t')] \rightarrow (PATH_{\leq}(s,t)$

$\wedge \forall z'((z' \neq s \wedge z' \neq t \wedge PATH_{\leq}(s,z') \wedge PATH_{\leq}(z',t)) \rightarrow \neg P_X(z')))))$

"V_p preserves $\leq^I$ restricted to the X in the quantifier prefix, and E_t."

(2)$\forall x,y_1,y_2,z_1,z_2({}_1({}_2(H_\phi(x,y_1,y_2) \wedge H_\phi(x,z_1,z_2)) \rightarrow ((y_1 = z_1 \wedge y_2 = z_2)$

$\wedge \exists x'(P_($$(x') \wedge PATH_{\leq}(x',x)) \wedge C(y_1,y_2)))_2$

"H_ϕ is a function: quantifier free part of $\varphi \rightarrow C$"

$\wedge ({}_2H_\phi(x,y_1,y_2) \wedge H_\phi(z,y_1,y_2) \rightarrow x=z)_2$ "H_ϕ is injective"

$\wedge \forall y_1',y_2',z_1',z_2',t_1',t_2',v',v_2$

$({}_2({}_3ST(v') \wedge \neg\exists y(E_{ST}(y,v'))$ "v' is the root in $<ST,E_{ST}>$"

$\wedge E_{ST}(v',v_2) \wedge M(v',y_1',y_2') \wedge M(v_2,z_1',z_2') \wedge E_C(t_1',t_2',z_1',z_2')$

$\wedge PATH_{E_C}(y_1',y_2',y_1,y_2) \wedge PATH_{E_C}(y_1,y_2,t_1',t_2'))_3 \rightarrow \exists x'(H_\phi(x',y_1,y_2)))_2$

"the range of H_ϕ is the first formula in $<C,E_C>$"

$\wedge (\exists x'(P_($$(x') \wedge PATH_{\leq}(x',x)) \rightarrow \exists y_1',y_2'(H_\phi(x,y_1',y_2')))_2)_1$

"the domain of H_ϕ is the quantifier free part of φ"

$\wedge \forall x,x',z,y_1,y_2,z_1,z_2[((H_\phi(x,y_1,y_2) \wedge H_\phi(z,z_1,z_2) \wedge E_C(y_1,y_2,z_1,z_2)) \rightarrow$

$(SUC_{\leq}(x,z) \vee (PATH_{\leq}(x,z) \wedge \forall x'((PATH_{\leq}(x,x') \wedge PATH_{\leq}(x',z)$

$\wedge x' \neq x \wedge x' \neq z) \rightarrow P_|(x'))))$

$\wedge (H_\phi(x,y_1,y_2) \rightarrow ((P_($$(x) \wedge C_($$(y_1,y_2)) \vee (P_)(x) \wedge C_)(y_1,y_2))$

$\vee (P_\wedge(x) \wedge C_\wedge(y_1,y_2)) \vee (P_\vee(x) \wedge C_\vee(y_1,y_2))$

$\vee (P_\neg(x) \wedge C_\neg(y_1,y_2)) \vee (P_X(x) \wedge (C_0(y_1,y_2) \vee C_1(y_1,y_2)))))]$

"H_ϕ preserves $\leq_I$ (ignoring |) and E_C, $\{\wedge,\vee,(,),\neg\}$, and maps X to 0/1"

(3) $\equiv \forall V_o^2 \exists z_0,y_0,z_f,y_f,z_0',y_0',z_f',y_f'[_1[_2$**(a)** $\wedge$ **(b)** $\wedge$ **(c)**$]_2 \rightarrow$ **(d)**$]_1$

(a) $[_1[_2(_3 V_o(z_0, y_0) \wedge \neg\exists z', y'(PRED_{\leq}(z_0, z') \wedge V_o(z', y'))$
$\wedge V_o(z_f, y_f) \wedge \neg\exists z', y'(SUC_{\leq}(z_f, z') \wedge V_o(z', y'))$

$\wedge \forall z'([PATH_{\leq}(z_0, z') \wedge PATH_{\leq}(z', z_f)] \rightarrow \exists y'(V_0(z', y')))$ total

$\wedge \forall y'([PATH_{\leq}(y_0, y') \wedge PATH_{\leq}(y', y_f)] \rightarrow \exists z'(V_0(z', y'))))_3$ surjective

"z_0 is the root in $dom(V_o)$, z_f is the leaf in $dom(V_o)$, y_0 is the root in $ran(V_o)$, y_f is the leaf in $ran(V_o)$"

(b) $\wedge \forall x, y, v, w[_3(V_o(x, y) \rightarrow (P_|(x) \wedge P_|(y)))$
$\wedge ((V_o(x, y) \wedge V_o(x, v)) \rightarrow y = v)$ function

$\wedge ((V_o(x, y) \wedge V_o(w, y)) \rightarrow x = w)$ injective

$\wedge ((V_o(x, y) \wedge V_o(v, w) \wedge SUC_{\leq}(x, v)) \rightarrow SUC_{\leq}(y, w)))]_3$
"V_o is a bijection from "|...|" in "QX |...|" to "|...|" in "(...X |...|...)", which also preserves $\leq^I$ "

(c) $\wedge (_3 PRED_{\leq}(z_0, z_0') \wedge P_X(z_0')$
"z_0' is the predecessor of the root in $dom(V_o)$, i.e., it is the X in the quantifier prefix"

$\wedge PRED_{\leq}(y_0, y_0') \wedge P_X(y_0')$
"y_0' is the predecessor of the root in $ran(V_o)$, i.e., it is the X in the quantifier free part"

$\wedge SUC_{\leq}(z_f, z_f') \wedge \neg P_|(z_f')$ "z_f' is the successor of the leaf in $dom(V_o)$"

$\wedge SUC_{\leq}(y_f, y_f') \wedge \neg P_|(y_f'))_3]_2$ "y_f' is the successor of the leaf in $ran(V_o)$"

(d) $\forall x, x'[_2(V_p(z_0', x) \wedge B_t(x, x')) \rightarrow (\exists z_1, z_2(H_\phi(y_0', z_1, z_2)$
$\wedge [(x' = "0" \wedge C_0(z_1, z_2)) \vee (x' = "1" \wedge C_1(z_1, z_2))]))]_2]_1$
"$B_t(V_p(z_0')) = H_\phi(y_0')$"

5.2.1.2.2 Expressing (2.2b): "… such that the formula ϕ is true" Evaluating the formula ϕ.

Once we have built the quantifier-free formula ϕ on {0, 1}, we must evaluate ϕ. We do so by evaluating one connective at a time, and one pair of matching parenthesis at a time, until the final result becomes 1.

Before we write the informal expression, let us look at an example of an evaluation process first. In this example, we can see that there are 10 evaluation steps, which correspond to 10 "operators" (i.e., either connectives or pairs of matching parenthesis) that are evaluated during the whole process. If there are at most n symbols in ϕ, that means that we need at most n evaluation steps to get the result of the formula ϕ. That is why the nodes of the graph (C, E_C) are pairs, and its edges are quadruples. In this way we allow the whole evaluation process to take up to n steps (where n is the length of the input formula φ), where in each step we have a propositional formula on {0, 1} with up to n symbols. Each node in the graph (ST, E_{ST}) represents one such formula, and the function M (for Marker) is a pointer which tells us in which node in (C, E_C) that formula begins **(*Figure 12*).** Note that in each evaluation step either one or two symbols are removed from the formula at the previous step.

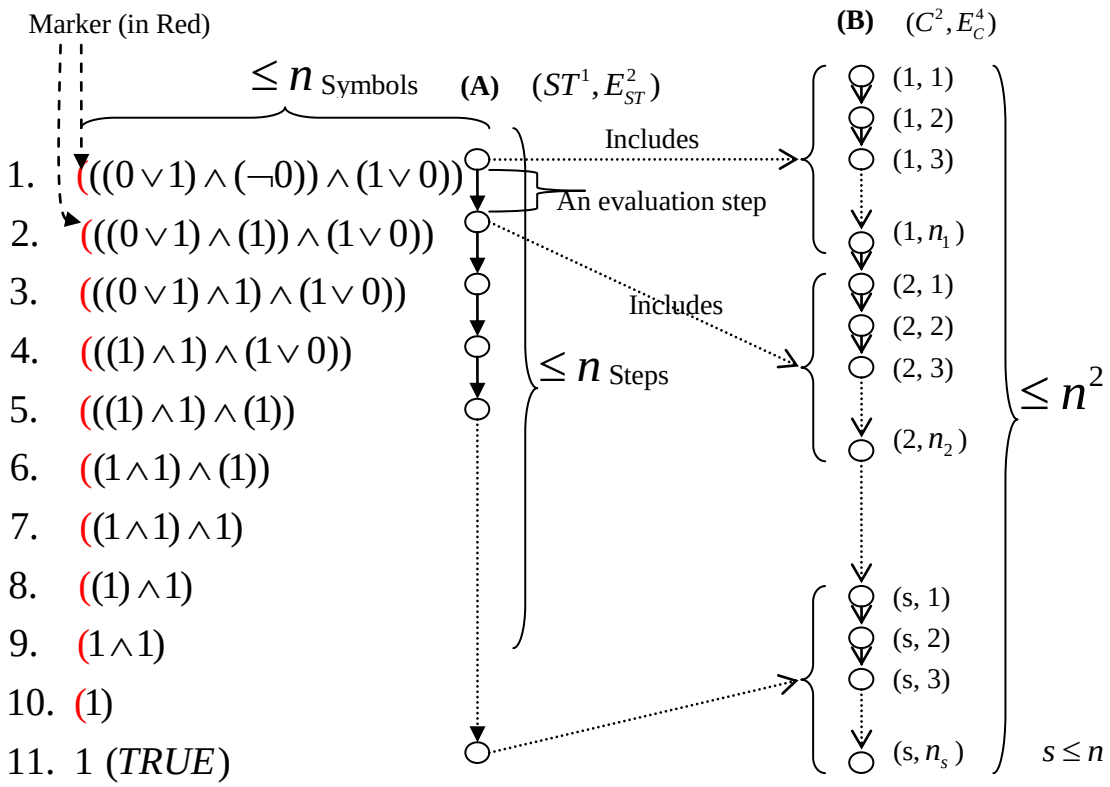


Figure 12

We redraw below the graphs (**A**) and (**B**) of ***Figure 12*** with a horizontal orientation (***Figure 13).*** Each evaluation step is called a stage. And the first symbol in each stage is given by the marker function M.

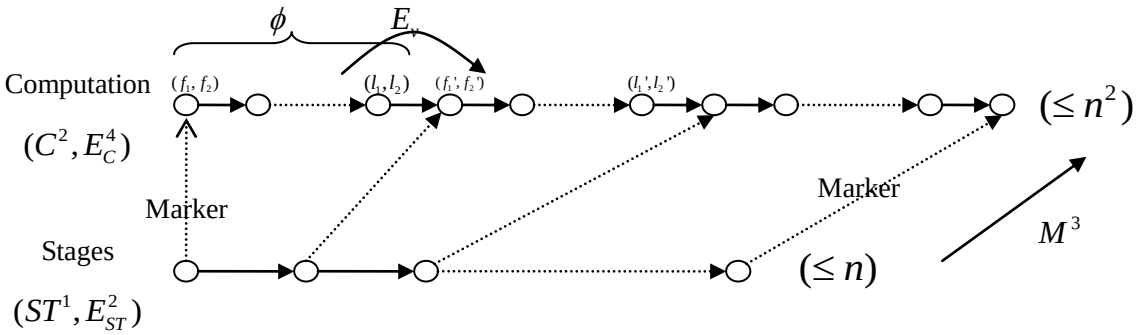


Figure 13

Based upon ***Figures 12 and 13***, we show below the steps we need to follow to express (2.2b).

["(C, E_C) is a linear graph"] **(1)**

$\wedge$["(ST, E_{ST}) is a linear graph"] **(2)**

$\wedge$["$M : ST^1 \to C^2$ is an injective and total function that preserves $PATH$ in E_{ST} and E_C"] **(3)**

$\wedge$["$C^2_\wedge, C^2_\vee, C^2_\neg, C^2_(, C^2_), C^2_1, C^2_0$ are pairwise disjoint, and $C^2_\wedge \bigcup C^2_\vee \bigcup C^2_\neg \bigcup C^2_(\bigcup C^2_) \bigcup C^2_1 \bigcup C^2_0 = C$"] **(4)**

$\wedge$["For every stage *x*, from stage *x* to stage *x*+1, we need to follow the rules of evaluation (See **Figure 12** Part A). The formula in $<C, E_C>$ at stage *x*+1 is the same as the formula at stage *x*, except for one of three possible sorts of changes, which correspond to the cases (**a**), (**b**) and (**c**) of **Figure 14**"] **(5)**

(Level 4) The respective formal expressions are showed below:

(1) $\equiv LINEAR^{2,4}(C, E_C)$

(2) $\equiv LINEAR(ST, E_{ST})$

(3) $({}_0 \forall s, s', t_1, t_2, k_1, k_2 ({}_1({}_2({}_3 M(s, t_1, t_2) \wedge M(s, k_1, k_2))_3$
$\to ({}_3(t_1 = k_1 \wedge t_2 = k_2) \wedge ST(s) \wedge C(t_1, t_2))_3)_2$
"M is a function, $M{:}ST^1 \to C^2$"

$\wedge ({}_3[M(s,k_1,k_2) \wedge M(t_1,k_1,k_2)] \to s{=}t_1)_3$ "M is injective"

$\wedge ({}_3 ST(s) \to \exists t_1', t_2' ({}_4 M(s, t_1', t_2'))_4)_3$ "M is total"

$\wedge ({}_3[{}_4 M(s,t_1,t_2) \wedge M(s', k_1,k_2) \wedge PATH_{ST}(s, s')]_4$
$\to PATH^{2,4}_{E_C}(t_1, t_2, k_1, k_2))_3)_1)_0$
"M preserves $PATH$ in E_{ST} and E_C"

(4) $\wedge \forall s_1, s_2((C_\wedge(s_1,s_2) \leftrightarrow \neg C_\vee(s_1,s_2)) \wedge (C_\wedge(s_1,s_2) \leftrightarrow \neg C_\neg(s_1,s_2))$
$\wedge (C_\wedge(s_1,s_2) \leftrightarrow \neg C_(\,(s_1,s_2)) \wedge (C_\wedge(s_1,s_2) \leftrightarrow \neg C_)(s_1,s_2))$
$\wedge (C_\wedge(s_1,s_2) \leftrightarrow \neg C_1(s_1,s_2)) \wedge (C_\wedge(s_1,s_2) \leftrightarrow \neg C_0(s_1,s_2)) \wedge ...)$
"$C^2_\wedge, C^2_\vee, C^2_\neg, C^2_(, C^2_), C^2_1, C^2_0$ are pairwise disjoint"

$\wedge \forall s_1, s_2(C(s_1,s_2) \rightarrow (C_\wedge(s_1,s_2) \vee C_\vee(s_1,s_2) \vee C_\neg(s_1,s_2) \vee C_(\,(s_1,s_2) \vee C_)(s_1,s_2)$
$\vee C_0(s_1,s_2) \vee C_1(s_1,s_2)))$

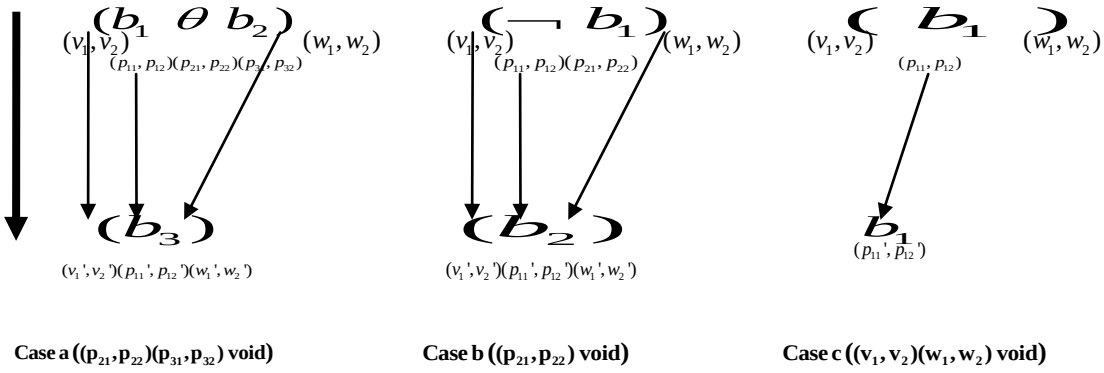
$\wedge \forall s_1, s_2((C_\wedge(s_1,s_2) \rightarrow C(s_1,s_2)) \wedge (C_\vee(s_1,s_2) \rightarrow C(s_1,s_2)) \wedge (C_\neg(s_1,s_2) \rightarrow C(s_1,s_2))$
$\wedge (C_(\,(s_1,s_2) \rightarrow C(s_1,s_2)) \wedge (C_)(s_1,s_2) \rightarrow C(s_1,s_2)) \wedge (C_0(s_1,s_2) \rightarrow C(s_1,s_2))$
$\wedge (C_1(s_1,s_2) \rightarrow C(s_1,s_2)))$
"$(C^2_\wedge \bigcup C^2_\vee \bigcup C^2_\neg \bigcup C^2_(\bigcup C^2_) \bigcup C^2_1 \bigcup C^2_0) = C$"

(5) $\equiv \forall x[ST(x) \rightarrow \exists E^4_v, f_1, f_2, l_1, l_2, f_1', f_2', l_1', l_2'$**[(d)** $\vee$ **(e)** $\vee$ **[(0)** $\wedge$ **((a)** $\vee$ **(b)** $\vee$ **(c))]]]**

The function E_v maps the formula at stage x to the formula at stage x+1.

The sub formula (d) corresponds to the last transition, i.e. the transition to the last formula in $< C, E_C >$ (“0” or “1”). The sub formula (e) corresponds to the last formula in $< C, E_C >$. The three sub formulas (a), (b), (c) correspond to the three possible cases (a), (b), (c) as in ***Figure 14***, according to which sort of operation is the one involved in the transition from the formula in stage x to the next formula in $< C, E_C >$. Note that the transition to the last formula (d) is necessarily an instance of case (c) in ***Figure 14***. For case (c) in ***Figure 14***, E_v is not total in its domain, since (v_1, v_2) (( ) and (w_1, w_2) ( )) are not mapped. For the last formula, E_v is not injective, since $(f_1', f_2')'' = ''(l_1', l_2')$ (*i.e.*, $f_1' = l_1'$ and $f_2' = l_2'$) (see ***Figure 16***).

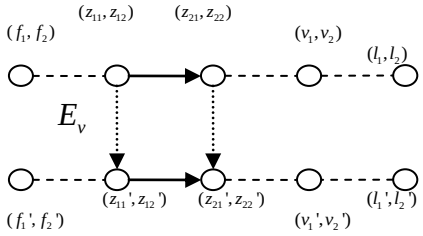
Function: E_v



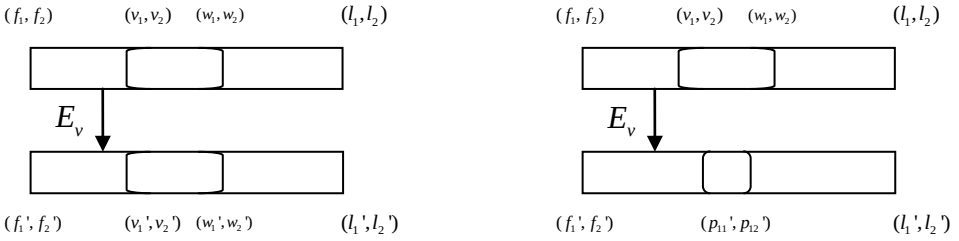
Case a ($(p_{21},p_{22})(p_{31},p_{32})$ void) **Case b ((p_{21},p_{22}) void)** **Case c ($(v_1,v_2)(w_1,w_2)$ void)**

Figure 14

Where $\theta \in \{\wedge, \vee\}$ and $b_1, b_2, b_3 \in \{0,1\}$.



Left side of the window



Cases a, b Case c

E_v Preserves E_C

Figure 15

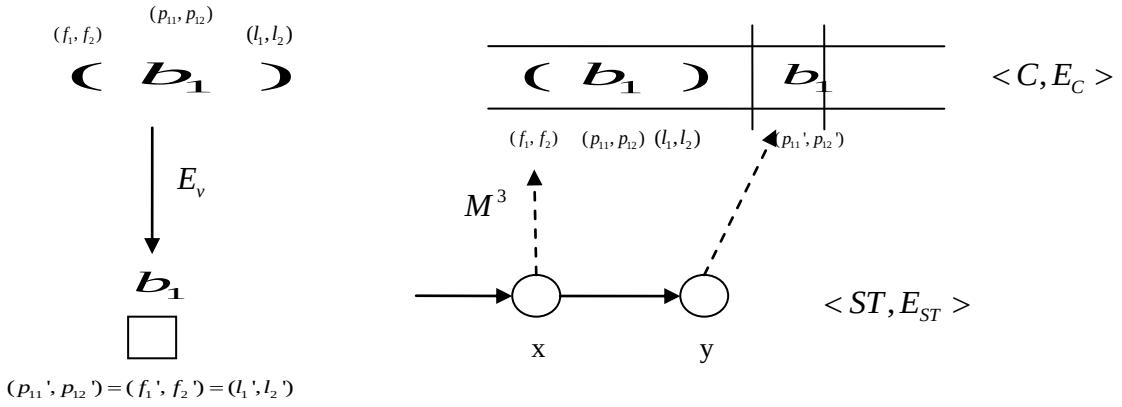


Figure 16

(0) $(_6(_7(_8\exists y, y_1(E_{ST}(x, y) \wedge E_{ST}(y, y_1))$

"x is not the leaf in E_{ST} and is not the predecessor of the leaf "

$$\wedge(_9\forall s_1, s_2, t_1, t_2, k_1, k_2(_0(_1(_2E_v(s_1, s_2, t_1, t_2) \wedge E_v(s_1, s_2, k_1, k_2))_2$$
$$\rightarrow(_2(t_1 = k_1 \wedge t_2 = k_2) \wedge C(s_1, s_2) \wedge C(t_1, t_2))_2)_1$$

"E_v is a function, E_v:$C \rightarrow C$ "

$$\wedge(_2[E_v(s_1, s_2, k_1, k_2) \wedge E_v(t_1, t_2, k_1, k_2)] \rightarrow (s_1 = t_1 \wedge s_2 = t_2))_2)_0)_9)_8)_7$$

"E_v is injective, E_v:C $\rightarrow C$ "

"$\exists$partial injection E_v^4 mapping the formula in $<C, E_C>$ in stage x to the formula in $<C, E_C>$ in stage $E_{ST}(x)$)"

$$\wedge(_7(M(x, f_1, f_2) \wedge "M(E_{ST}(x), E_C(l_1, l_2))" \wedge "E_C(l_1, l_2) = (f_1', f_2')"$$
$$\wedge "E_C^{-1}(M[E_{ST}(E_{ST}(x))], l_1', l_2')")$$

"$[(f_1, f_2), (l_1, l_2)]$ and $[(f_1', f_2'), (l_1', l_2')]$ are the **delimiters** of the two formulas as in Figure 15"

$$\wedge \forall y_1, y_2, z_1, z_2((E_v(y_1, y_2, z_1, z_2)) \rightarrow [((PATH_{E_C}^{2,4}(f_1, f_2, y_1, y_2) \wedge PATH_{E_C}^{2,4}(y_1, y_2, l_1, l_2)))$$
$$\wedge (PATH_{E_C}^{2,4}(f_1', f_2', z_1, z_2) \wedge PATH_{E_C}^{2,4}(z_1, z_2, l_1', l_2'))]$$
$$\wedge E_v(f_1, f_2, f_1', f_2') \wedge E_v(l_1, l_2, l_1', l_2'))_7)_6$$

"E_v maps nodes from the subgraph $[(f_1, f_2), (l_1, l_2)]$ to the subgraph $[(f_1', f_2'), (l_1', l_2')]$"

(a) $(_{-1}\exists v_1, v_2, w_1, w_2, v_1', v_2', w_1', w_2', p_{11}, p_{12}, p_{21}, p_{22}, p_{31}, p_{32}, p_{11}', p_{12}'$

$$(_0 PATH_{E_C}^{2,4}(f_1, f_2, v_1, v_2) \wedge PATH_{E_C}^{2,4}(w_1, w_2, l_1, l_2) \wedge E_C(p_{11}, p_{12}, p_{21}, p_{22})$$
$$\wedge E_C(p_{21}, p_{22}, p_{31}, p_{32}) \wedge E_C(v_1, v_2, p_{11}, p_{12}) \wedge E_C(p_{31}, p_{32}, w_1, w_2)$$
$$\wedge C_(v_1, v_2) \wedge C_)(w_1, w_2) \wedge PATH_{E_C}^{2,4}(f_1', f_2', v_1', v_2') \wedge PATH_{E_C}^{2,4}(w_1', w_2', l_1', l_2')$$
$$\wedge E_C(v_1', v_2', p_{11}', p_{12}') \wedge E_C(p_{11}', p_{12}', w_1', w_2') \wedge E_v(p_{11}, p_{12}, p_{11}', p_{12}')$$
$$\wedge E_v(v_1, v_2, v_1', v_2') \wedge E_v(w_1, w_2, w_1', w_2') \wedge C_(v_1', v_2') \wedge C_)(w_1', w_2')$$

"$[(v_1, v_2), (w_1, w_2)]$ and $[(v_1', v_2'), (w_1', w_2')]$ define the **windows of change** (Figure 15 a, b)"

$$\wedge\forall z_{11}, z_{12}, z_{21}, z_{22}, z_{11}', z_{12}', z_{21}', z_{22}'[_1(_2(_3 PATH^{2,4}_{E_C}(f_1, f_2, z_{11}, z_{12}) \wedge PATH^{2,4}_{E_C}(z_{21}, z_{22}, v_1, v_2)$$
$$\wedge E_C(z_{11}, z_{12}, z_{21}, z_{22}) \wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}') \wedge E_v(z_{21}, z_{22}, z_{21}', z_{22}'))_3$$
$$\rightarrow (_3 PATH^{2,4}_{E_C}(f_1', f_2', z_{11}', z_{12}') \wedge PATH^{2,4}_{E_C}(z_{21}', z_{22}', v_1', v_2') \wedge E_C(z_{11}', z_{12}', z_{21}', z_{22}'))_3)_2$$
$$\wedge (_2(_3 PATH^{2,4}_{E_C}(w_1, w_2, z_{11}, z_{12}) \wedge PATH^{2,4}_{E_C}(z_{21}, z_{22}, l_1, l_2) \wedge E_C(z_{11}, z_{12}, z_{21}, z_{22})$$
$$\wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}') \wedge E_v(z_{21}, z_{22}, z_{21}', z_{22}'))_3$$
$$\rightarrow (_3 PATH^{2,4}_{E_C}(w_1', w_2', z_{11}', z_{12}') \wedge PATH^{2,4}_{E_C}(z_{21}', z_{22}', l_1', l_2') \wedge E_C(z_{11}', z_{12}', z_{21}', z_{22}'))_3)_2]_1$$

"E_v preserves E_C outside of the windows, and preserves left and right side of the windows (Figure 15)"

$$\wedge\forall z_{11}, z_{12}, z_{11}', z_{12}'[_1(_2 PATH^{2,4}_{E_C}(f_1, f_2, z_{11}, z_{12}) \wedge PATH^{2,4}_{E_C}(z_{11}, z_{12}, v_1, v_2) \wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}'))_2$$
$$\rightarrow (_2 PATH^{2,4}_{E_C}(f_1', f_2', z_{11}', z_{12}') \wedge PATH^{2,4}_{E_C}(z_{11}', z_{12}', v_1', v_2')$$
$$\wedge [_3(\mathrm{C}_{(}(z_{11}, z_{12}) \wedge \mathrm{C}_{(}(z_{11}', z_{12}')) \vee (\mathrm{C}_{)}(z_{11}, z_{12}) \wedge \mathrm{C}_{)}(z_{11}', z_{12}'))$$
$$\vee (\mathrm{C}_{\wedge}(z_{11}, z_{12}) \wedge \mathrm{C}_{\wedge}(z_{11}', z_{12}')) \vee (\mathrm{C}_{\vee}(z_{11}, z_{12}) \wedge \mathrm{C}_{\vee}(z_{11}', z_{12}'))$$
$$\vee (\mathrm{C}_0(z_{11}, z_{12}) \wedge \mathrm{C}_0(z_{11}', z_{12}')) \vee (\mathrm{C}_1(z_{11}, z_{12}) \wedge \mathrm{C}_1(z_{11}', z_{12}'))$$
$$\vee (\mathrm{C}_{\neg}(z_{11}, z_{12}) \wedge \mathrm{C}_{\neg}(z_{11}', z_{12}'))]_3)_2]_1$$

"E_v preserves symbols in **left side of the window**"

$$\wedge\forall z_{11}, z_{12}, z_{11}', z_{12}'[_1(_2 PATH^{2,4}_{E_C}(w_1, w_2, z_{11}, z_{12}) \wedge PATH^{2,4}_{E_C}(z_{11}, z_{12}, l_1, l_2)$$
$$\wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}'))_2 \rightarrow (_2 PATH^{2,4}_{E_C}(w_1', w_2', z_{11}', z_{12}') \wedge PATH^{2,4}_{E_C}(z_{11}', z_{12}', l_1', l_2')$$
$$\wedge [_3(\mathrm{C}_{(}(z_{11}, z_{12}) \wedge \mathrm{C}_{(}(z_{11}', z_{12}')) \vee (\mathrm{C}_{)}(z_{11}, z_{12}) \wedge \mathrm{C}_{)}(z_{11}', z_{12}'))$$
$$\vee (\mathrm{C}_{\wedge}(z_{11}, z_{12}) \wedge \mathrm{C}_{\wedge}(z_{11}', z_{12}')) \vee (\mathrm{C}_{\vee}(z_{11}, z_{12}) \wedge \mathrm{C}_{\vee}(z_{11}', z_{12}'))$$
$$\vee (\mathrm{C}_0(z_{11}, z_{12}) \wedge \mathrm{C}_0(z_{11}', z_{12}')) \vee (\mathrm{C}_1(z_{11}, z_{12}) \wedge \mathrm{C}_1(z_{11}', z_{12}'))$$
$$\vee (\mathrm{C}_{\neg}(z_{11}, z_{12}) \wedge \mathrm{C}_{\neg}(z_{11}', z_{12}'))]_3)_2]_1$$

"E_v preserves symbols in **right side of the window**"

$$\wedge [_1(C_0(p_{11}, p_{12}) \wedge C_0(p_{31}, p_{32}) \wedge C_{\wedge}(p_{21}, p_{22}) \wedge C_0(p_{11}', p_{12}'))$$
$$\vee (C_0(p_{11}, p_{12}) \wedge C_0(p_{31}, p_{32}) \wedge C_{\vee}(p_{21}, p_{22}) \wedge C_0(p_{11}', p_{12}'))$$
$$\vee (C_0(p_{11}, p_{12}) \wedge C_1(p_{31}, p_{32}) \wedge C_{\wedge}(p_{21}, p_{22}) \wedge C_0(p_{11}', p_{12}'))$$
$$\vee (C_0(p_{11}, p_{12}) \wedge C_1(p_{31}, p_{32}) \wedge C_{\vee}(p_{21}, p_{22}) \wedge C_1(p_{11}', p_{12}'))$$
$$\vee (C_1(p_{11}, p_{12}) \wedge C_0(p_{31}, p_{32}) \wedge C_{\wedge}(p_{21}, p_{22}) \wedge C_0(p_{11}', p_{12}'))$$
$$\vee (C_1(p_{11}, p_{12}) \wedge C_0(p_{31}, p_{32}) \wedge C_{\vee}(p_{21}, p_{22}) \wedge C_1(p_{11}', p_{12}'))$$
$$\vee (C_1(p_{11}, p_{12}) \wedge C_1(p_{31}, p_{32}) \wedge C_{\wedge}(p_{21}, p_{22}) \wedge C_1(p_{11}', p_{12}'))$$
$$\vee (C_1(p_{11}, p_{12}) \wedge C_1(p_{31}, p_{32}) \wedge C_{\vee}(p_{21}, p_{22}) \wedge C_1(p_{11}', p_{12}'))]_1)_0)_{-1}$$

"In (p_{11}', p_{12}') we get the **result** of applying the operator θ in (p_{21}, p_{22}) to the Boolean values b_1, in (p_{11}, p_{12}), and b_2, in (p_{31}, p_{32}) (Figure 14a)"

(b) $({}_{-1}\exists v_1, v_2, w_1, w_2, v_1', v_2', w_1', w_2', p_{11}, p_{12}, p_{21}, p_{22}, p_{11}', p_{12}'$

$({}_0 PATH_{E_C}^{2,4}(f_1, f_2, v_1, v_2) \wedge PATH_{E_C}^{2,4}(w_1, w_2, l_1, l_2) \wedge E_C(p_{11}, p_{12}, p_{21}, p_{22})$

$\wedge E_C(v_1, v_2, p_{11}, p_{12}) \wedge E_C(p_{21}, p_{22}, w_1, w_2)$

$\wedge C_(v_1, v_2) \wedge C_)(w_1, w_2) \wedge PATH_{E_C}^{2,4}(f_1', f_2', v_1', v_2') \wedge PATH_{E_C}^{2,4}(w_1', w_2', l_1', l_2')$

$\wedge E_C(v_1', v_2', p_{11}', p_{12}') \wedge E_C(p_{11}', p_{12}', w_1', w_2') \wedge E_v(p_{11}, p_{12}, p_{11}', p_{12}')$

$\wedge E_v(v_1, v_2, v_1', v_2') \wedge E_v(w_1, w_2, w_1', w_2') \wedge C_(v_1', v_2') \wedge C_)(w_1', w_2')$

"$[(v_1, v_2), (w_1, w_2)]$ and $[(v_1', v_2'), (w_1', w_2')]$ define the windows of change (Figure 15 a, b)"

$\wedge \forall z_{11}, z_{12}, z_{21}, z_{22}, z_{11}', z_{12}', z_{21}', z_{22}' [_1({}_2({}_3 PATH_{E_C}^{2,4}(f_1, f_2, z_{11}, z_{12}) \wedge PATH_{E_C}^{2,4}(z_{21}, z_{22}, v_1, v_2)$

$\wedge E_C(z_{11}, z_{12}, z_{21}, z_{22}) \wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}') \wedge E_v(z_{21}, z_{22}, z_{21}', z_{22}'))_3$

$\rightarrow ({}_3 PATH_{E_C}^{2,4}(f_1', f_2', z_{11}', z_{12}') \wedge PATH_{E_C}^{2,4}(z_{21}', z_{22}', v_1', v_2') \wedge E_C(z_{11}', z_{12}', z_{21}', z_{22}'))_3)_2$

$\wedge ({}_2({}_3 PATH_{E_C}^{2,4}(w_1, w_2, z_{11}, z_{12}) \wedge PATH_{E_C}^{2,4}(z_{21}, z_{22}, l_1, l_2) \wedge E_C(z_{11}, z_{12}, z_{21}, z_{22})$

$\wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}') \wedge E_v(z_{21}, z_{22}, z_{21}', z_{22}'))_3$

$\rightarrow ({}_3 PATH_{E_C}^{2,4}(w_1', w_2', z_{11}', z_{12}') \wedge PATH_{E_C}^{2,4}(z_{21}', z_{22}', l_1', l_2') \wedge E_C(z_{11}', z_{12}', z_{21}', z_{22}'))_3)_2]_1$

"E_v preserves E_C outside of the windows, and preserves left and right side of the windows (Figure 15)"

$\wedge \forall z_{11}, z_{12}, z_{11}', z_{12}' [_1({}_2 PATH_{E_C}^{2,4}(f_1, f_2, z_{11}, z_{12}) \wedge PATH_{E_C}^{2,4}(z_{11}, z_{12}, v_1, v_2)$

$\wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}'))_2 \rightarrow ({}_2 PATH_{E_C}^{2,4}(f_1', f_2', z_{11}', z_{12}') \wedge PATH_{E_C}^{2,4}(z_{11}', z_{12}', v_1', v_2')$

$\wedge [_3(\mathrm{C}_(z_{11}, z_{12}) \wedge \mathrm{C}_(z_{11}', z_{12}')) \vee (\mathrm{C}_)(z_{11}, z_{12}) \wedge \mathrm{C}_)(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_\wedge(z_{11}, z_{12}) \wedge \mathrm{C}_\wedge(z_{11}', z_{12}')) \vee (\mathrm{C}_\vee(z_{11}, z_{12}) \wedge \mathrm{C}_\vee(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_0(z_{11}, z_{12}) \wedge \mathrm{C}_0(z_{11}', z_{12}')) \vee (\mathrm{C}_1(z_{11}, z_{12}) \wedge \mathrm{C}_1(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_\neg(z_{11}, z_{12}) \wedge \mathrm{C}_\neg(z_{11}', z_{12}'))]_3)_2]_1$

"E_v preserves symbols in left side of the window"

$\wedge \forall z_{11}, z_{12}, z_{11}', z_{12}' [_1({}_2 PATH_{E_C}^{2,4}(w_1, w_2, z_{11}, z_{12}) \wedge PATH_{E_C}^{2,4}(z_{11}, z_{12}, l_1, l_2)$

$\wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}'))_2 \rightarrow ({}_2 PATH_{E_C}^{2,4}(w_1', w_2', z_{11}', z_{12}') \wedge PATH_{E_C}^{2,4}(z_{11}', z_{12}', l_1', l_2')$

$\wedge [_3(\mathrm{C}_(z_{11}, z_{12}) \wedge \mathrm{C}_(z_{11}', z_{12}')) \vee (\mathrm{C}_)(z_{11}, z_{12}) \wedge \mathrm{C}_)(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_\wedge(z_{11}, z_{12}) \wedge \mathrm{C}_\wedge(z_{11}', z_{12}')) \vee (\mathrm{C}_\vee(z_{11}, z_{12}) \wedge \mathrm{C}_\vee(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_0(z_{11}, z_{12}) \wedge \mathrm{C}_0(z_{11}', z_{12}')) \vee (\mathrm{C}_1(z_{11}, z_{12}) \wedge \mathrm{C}_1(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_\neg(z_{11}, z_{12}) \wedge \mathrm{C}_\neg(z_{11}', z_{12}'))]_3)_2]_1$

"E_v preserves symbols in right side of the window"

$\wedge [_1(C_0(p_{21}, p_{22}) \wedge C_\neg(p_{11}, p_{12}) \wedge C_1(p_{11}', p_{12}'))$

$\vee (C_1(p_{21}, p_{22}) \wedge C_\neg(p_{11}, p_{12}) \wedge C_0(p_{11}', p_{12}'))]_1)_0)_{-1}$

"In (p_{11}', p_{12}') we get the result of applying $\neg$, in (p_{11}, p_{12}), to the Boolean value b_1, in (p_{21}, p_{22}) (Figure 14b)"

(c) $({}_{-1}\exists v_1, v_2, w_1, w_2, p_{11}, p_{12}, p_{11}', p_{12}'$

$({}_0 PATH^{2,4}_{E_C}(f_1, f_2, v_1, v_2) \wedge PATH^{2,4}_{E_C}(w_1, w_2, l_1, l_2) \wedge E_C(v_1, v_2, p_{11}, p_{12}) \wedge E_C(p_{11}, p_{12}, w_1, w_2)$

$\wedge C_{(}(v_1, v_2) \wedge C_{)}(w_1, w_2) \wedge PATH^{2,4}_{E_C}(f_1', f_2', p_{11}', p_{12}') \wedge PATH^{2,4}_{E_C}(p_{11}', p_{12}', l_1', l_2')$

$\wedge E_v(p_{11}, p_{12}, p_{11}', p_{12}')$

"$[(v_1, v_2), (w_1, w_2)]$ and $[(p_{11}', p_{12}'), (p_{11}', p_{12}')]$ define the windows of change (Figure 15c)"

$\wedge \forall z_{11}, z_{12}, z_{21}, z_{22}, z_{11}', z_{12}', z_{21}', z_{22}' [_1 ({}_2 ({}_3 PATH^{2,4}_{E_C}(f_1, f_2, z_{11}, z_{12}) \wedge PATH^{2,4}_{E_C}(z_{21}, z_{22}, v_1, v_2)$

$\wedge E_C(z_{11}, z_{12}, z_{21}, z_{22}) \wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}') \wedge E_v(z_{21}, z_{22}, z_{21}', z_{22}'))_3$

$\rightarrow ({}_3 PATH^{2,4}_{E_C}(f_1', f_2', z_{11}', z_{12}') \wedge E_C(z_{11}', z_{12}', z_{21}', z_{22}') \wedge PATH^{2,4}_{E_C}(z_{21}', z_{22}', p_{11}', p_{12}'))_3)_2$

$\wedge ({}_2 ({}_3 PATH^{2,4}_{E_C}(w_1, w_2, z_{11}, z_{12}) \wedge PATH^{2,4}_{E_C}(z_{21}, z_{22}, l_1, l_2) \wedge E_C(z_{11}, z_{12}, z_{21}, z_{22})$

$\wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}') \wedge E_v(z_{21}, z_{22}, z_{21}', z_{22}'))_3$

$\rightarrow ({}_3 PATH^{2,4}_{E_C}(z_{21}', z_{22}', l_1', l_2') \wedge E_C(z_{11}', z_{12}', z_{21}', z_{22}') \wedge PATH^{2,4}_{E_C}(p_{11}', p_{12}', z_{11}', z_{12}'))_3)_2]_1$

"E_v preserves E_C outside of the windows, and preserves left and right side of the windows (Figure 15)"

$\wedge \forall z_{11}, z_{12}, z_{11}', z_{12}' [_1 ({}_2 PATH^{2,4}_{E_C}(f_1, f_2, z_{11}, z_{12}) \wedge PATH^{2,4}_{E_C}(z_{11}, z_{12}, v_1, v_2)$

$\wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}'))_2 \rightarrow ({}_2 PATH^{2,4}_{E_C}(f_1', f_2', z_{11}', z_{12}') \wedge PATH^{2,4}_{E_C}(z_{21}', z_{22}', p_{11}', p_{12}')$

$\wedge [_3 (C_{(}(z_{11}, z_{12}) \wedge C_{(}(z_{11}', z_{12}')) \vee (C_{)}(z_{11}, z_{12}) \wedge C_{)}(z_{11}', z_{12}'))$

$\vee (C_{\wedge}(z_{11}, z_{12}) \wedge C_{\wedge}(z_{11}', z_{12}')) \vee (C_{\vee}(z_{11}, z_{12}) \wedge C_{\vee}(z_{11}', z_{12}'))$

$\vee (C_0(z_{11}, z_{12}) \wedge C_0(z_{11}', z_{12}')) \vee (C_1(z_{11}, z_{12}) \wedge C_1(z_{11}', z_{12}'))$

$\vee (C_{\neg}(z_{11}, z_{12}) \wedge C_{\neg}(z_{11}', z_{12}'))]_3)_2]_1$

"E_v preserves symbols in left side of the window"

$\wedge \forall z_{11}, z_{12}, z_{11}', z_{12}' [_1 ({}_2 PATH^{2,4}_{E_C}(w_1, w_2, z_{11}, z_{12}) \wedge PATH^{2,4}_{E_C}(z_{11}, z_{12}, l_1, l_2)$

$\wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}'))_2 \rightarrow ({}_2 PATH^{2,4}_{E_C}(p_{11}', p_{12}', z_{11}', z_{12}') \wedge PATH^{2,4}_{E_C}(z_{11}', z_{12}', l_1', l_2')$

$\wedge [_3 (C_{(}(z_{11}, z_{12}) \wedge C_{(}(z_{11}', z_{12}')) \vee (C_{)}(z_{11}, z_{12}) \wedge C_{)}(z_{11}', z_{12}'))$

$\vee (C_{\wedge}(z_{11}, z_{12}) \wedge C_{\wedge}(z_{11}', z_{12}')) \vee (C_{\vee}(z_{11}, z_{12}) \wedge C_{\vee}(z_{11}', z_{12}'))$

$\vee (C_0(z_{11}, z_{12}) \wedge C_0(z_{11}', z_{12}')) \vee (C_1(z_{11}, z_{12}) \wedge C_1(z_{11}', z_{12}'))$

$\vee (C_{\neg}(z_{11}, z_{12}) \wedge C_{\neg}(z_{11}', z_{12}'))]_3)_2]_1$

"E_v preserves symbols in right side of the window"

$\wedge [_1 (C_0(p_{11}, p_{12}) \wedge C_0(p_{11}', p_{12}')) \vee (C_1(p_{11}, p_{12}) \wedge C_1(p_{11}', p_{12}'))]_1)_0)_{-1}$

"In (p_{11}', p_{12}') we get a copy of the Boolean value b_1, in (p_{11}, p_{12}) (Figure 14c)"

(d) $({}_0 \exists y({}_1 E_{ST}(x,y) \wedge \neg\exists z(E_{ST}(y,z))$
$\wedge \exists p_{11}, p_{12}, p_{11}', p_{12}'({}_2 M(x, f_1, f_2) \wedge M(y, p_{11}', p_{12}') \wedge E_C(f_1, f_2, p_{11}, p_{12})$
$\wedge E_C(p_{11}, p_{12}, l_1, l_2) \wedge E_C(l_1, l_2, p_{11}', p_{12}')$
$\wedge \neg\exists p_{21}', p_{22}'(E_C(p_{11}', p_{12}', p_{21}', p_{22}'))$
$\wedge C_(f_1, f_2) \wedge C_)(l_1, l_2) \wedge [{}_3(C_1(p_{11}, p_{12}) \wedge C_1(p_{11}', p_{12}'))$
$\vee (C_0(p_{11}, p_{12}) \wedge C_0(p_{11}', p_{12}'))]_3)_2)_1)_0$

"x is the predecessor of the leaf in E_{ST}, so that this is the last transition" (See Figure 16)
Note that the last transition is necessarily an instance of case c in Figure 14.

(e) $({}_0 \neg\exists y(E_{ST}(x,y))$ "x is the leaf in E_{ST}"
$\wedge \exists p_1', p_2'(M(x, p_1', p_2') \wedge \neg\exists y_1', y_2'(E_C(p_1', p_2', y_1', y_2'))$ "(p_1', p_2') is a leaf in E_C"
$\wedge C_1(p_1', p_2'))))_0$
"the last formula in $<C, E_C>$ is "1" "

5.2.1.3 Complete expression for $SATQBF_k$

We get the complete expression by combining the three parts together (that is taking (2.1) from 5.2.1.1, (2.2a) from 5.2.1.2.1, and (2.2b) from 5.2.1.2.2, together with the quantification in 5.2.1.2).

$\equiv \exists V_1^1, E_1^2, B_1^2, \forall V_2^1, E_2^2, B_2^2,$
$\exists ..., Q_k V_k^1, E_k^2, B_k^2, \exists V_t^1, E_t^2, B_t^2, V_p^2, \exists C^2, E_c^4, ST^1, E_{ST}^2, M^3,$
$C_\wedge^2, C_\vee^2, C_\neg^2, C_(^2, C_)^2, C_1^2, C_0^2, H_\phi^3$

$({}_0$["$G_t = <V_t^1, E_t^2>$ is a linear graph"] **(1)**

$\wedge$**[**"The length of G_t is equal to the number of variables in the quantifier prefix of φ"] **(2)**

$\wedge$["$G_1 = <V_1, E_1>, G_3 = <V_3, E_3>, ..., G_{k'} = <V_{k'}, E_{k'}>$ are linear graphs"] **(3)**
(where k' is the index of the last $\exists$ quantifier in the prefix)

$\wedge$["$B_1 : V_1 \to \{0,1\}, B_3 : V_3 \to \{0,1\}, ..., B_{k'} : V_{k'} \to \{0,1\}$ are total functions"] **(4)**

$\wedge$["the lengths of the linear graphs $G_1, G_3, ..., G_{k'}$ are equal to the lengths of the corresponding quantifier blocks in φ"] **(5)**

$\wedge$["V_p is a partial bijection from the quantifier prefix of φ (restricted to the X in the quantifier prefix) to V_t, which maps every X to its corresponding node in G_t, and which preserves $\leq^I$ and E_t."] **(6)**

$\wedge$["(C, E_C) is a linear graph"] **(7)**

$\wedge$["(ST, E_{ST}) is a linear graph"] **(8)**

$\wedge$["$M : ST^1 \to C^2$ is an injective and total function that preserves $PATH$ in E_{ST} and E_C"] **(9)**

$\wedge$["$C^2_{\wedge}, C^2_{\vee}, C^2_{\neg}, C^2_{(}, C^2_{)}, C^2_1, C^2_0$ are pairwise disjoint sets, and $C^2_{\wedge} \bigcup C^2_{\vee} \bigcup C^2_{\neg} \bigcup C^2_{(} \bigcup C^2_{)} \bigcup C^2_1 \bigcup C^2_0 = C$"] **(10)**

$\wedge$["H_ϕ is a partial surjective injection from the quantifier free part of φ to the first formula in (C, E_C), which maps every "X" in the quantifier free part to the corresponding position in the first formula in (C, E_c) which we denote by ϕ, which preserves $\wedge, \vee, (,), \neg$, and $\leq_I$ and E_C, and which ignores "|". "] **(11)**

$\wedge$[$_1$

[$_2$(["$V_1, V_2, V_3, ..., V_k$ are disjoint sets"] **(12)**

$\wedge$["$G_2 =< V_2, E_2 >, G_4 =< V_4, E_4 > ... G_{k"} =< V_{k"}, E_{k"} >$ are linear graphs"] **(13)**
(where $k"$ is the index of the last $\forall$ quantifier in the prefix)

$\wedge$["$B_2 : V_2 \to \{0,1\}, B_4 : V_4 \to \{0,1\}, ..., B_{k"} : V_{k"} \to \{0,1\}$ are total functions"] **(14)**

$\wedge$["the lengths of the linear graphs $G_2, G_4, ..., G_{k"}$ are equal to the lengths of the corresponding quantifier blocks in φ"]) **(15)**

$\wedge \exists U^2_1, U^2_2 ... U^2_k$(["$U_1$ is a partial bijection: $G_1 \to$ 1st part of G_t"]
$\wedge$["U_2 is a partial bijection: $G_2 \to$ 2nd part of G_t"]
$\wedge$ ... ... $\wedge$["U_k is a partial bijection: $G_k \to k-$th part of G_t"]) **(16)**

$\wedge$["B_t:$V_t \rightarrow \{0,1\}$ is a total function that coincides with $B_1, B_2, ..., B_k$"
$\equiv$ "$B_t = B_1 \bigcup B_2 \bigcup ... \bigcup B_k$"]]$_2$ **(17)**

$\rightarrow$[$_2 \forall V_o^2 \exists z_0, y_0, z_f, y_f, z_0', y_0', z_f', y_f'$[$_3$($_4$
[$_5$"z_0 is the root in $dom(V_o)$, z_f is the leaf in $dom(V_o)$,
y_0 is the root in $ran(V_o)$, y_f is the leaf in $ran(V_o)$"]$_5$ **(18)**

$\wedge$[$_5$"V_o is a partial bijection from "|...|" in "QX |...|" to "|...|" in "(...X |...|...)",
which preserves $\leq_I$ "]$_5$ **(19)**

$\wedge$[$_5$"z_0' is predecessor of root in dom (V_o)"
$\wedge$"y_0' is predecessor of root in ran (V_o)"
$\wedge$"z_f' is successor of leaf in dom (V_o)"
$\wedge$"y_f' is successor of leaf in ran (V_o)"]$_5$)$_4$ **(20)**

$\rightarrow$($_4$["$B_t(V_p(z_0')) = H_\phi(y_0')$"])$_4$]$_3$ **(21)**

$\wedge$["The truth value of ϕ on the valuation B_t is 1"]]$_2$]$_1$)$_0$ **(22)**

(1) "The graph G_t is a linear graph": $LINEAR(V_t, E_t)$

(2) "The length of G_t is equal to the number of variables in the quantifier prefix of φ"

This statement is implied by the following statement:
"V_p is a partial bijection from the quantifier prefix of φ (restricted to the X in the quantifier prefix) to V_t, which maps every X to its corresponding node in G_t, and which preserves $\leq^I$ and E_t." (See (6) below)

(3) "$G_1 =< V_1, E_1 >, G_3 =< V_3, E_3 >, ..., G_{k'} =< V_{k'}, E_{k'} >$ are linear graphs":

(where k' is the index of the last $\exists$ quantifier in the prefix)

$LINEAR(V_1, E_1) \wedge LINEAR(V_3, E_3) \wedge ... \wedge LINEAR(V_{k'}, E_{k'})$

(4) "$B_1 : V_1 \to \{0,1\} \wedge B_3 : V_3 \to \{0,1\} \wedge ... \wedge B_{k'} : V_{k'} \to \{0,1\}$ are total functions"

$$\begin{aligned}
&\wedge[\forall t, p, p'[B_1(t,p) \wedge B_1(t,p') \to (p = p') && \text{"function"}\\
&\qquad \wedge (V_1(t) \to \exists p(B_1(t,p))) && \text{"total"}\\
&\qquad \wedge (B_1(t,p) \to (p = \text{"0"} \vee p = \text{"1"}))]\\
&\\
&\wedge \forall t, p, p'[B_3(t,p) \wedge B_3(t,p') \to (p = p') && \text{"function"}\\
&\qquad \wedge (V_3(t) \to \exists p(B_3(t,p))) && \text{"total"}\\
&\qquad \wedge (B_3(t,p) \to (p = \text{"0"} \vee p = \text{"1"}))]\\
&\wedge .....\\
&\wedge \forall t, p, p'[B_{k'}(t,p) \wedge B_{k'}(t,p') \to (p = p') && \text{"function"}\\
&\qquad \wedge (V_{k'}(t) \to \exists p(B_{k'}(t,p))) && \text{"total"}\\
&\qquad \wedge (B_{k'}(t,p) \to (p = \text{"0"} \vee p = \text{"1"}))]]
\end{aligned}$$

(5) If $k' \neq k,$ then (by $\alpha_{k'1}$ we mean the formula template α_{i1} instantiated with $i = k'$):

$$\bigwedge_{i=1,3,5,...,k'} \exists L'^2 v_1 v_2 v_3 ... v_{k'} v_{k'+1} (\alpha_{k'1} \wedge \alpha_{i2})$$

If $k' = k,$ then (by $\alpha_{k'-2,1}$ we mean the formula template α_{i1} instantiated with $i = k' - 2$):
($k' - 2$ is the previous to the last $\exists$ block, and the subformulas $\beta_1, \beta_2, \beta_3$
take care of the last quantifier block)

$$\left(\bigwedge_{i=1,3,5,...,k'-2} \exists L'^2 v_1 v_2 v_3 ... v_{k'-1} (\alpha_{k'-2,1} \wedge \alpha_{i2})\right) \wedge \exists L'^2 v_1 v_2 v_3 ... v_k v_e (\beta_1 \wedge \beta_2 \wedge \beta_3)$$

"for $1 \le j \le i,$ v_j is the first quantifier of the j-th alternating quantifier block:"
(when i is not the index of the last block)
$\alpha_{i1} \equiv ({}_0 P_\exists(v_1) \wedge P_\forall(v_2) \wedge P_\exists(v_3) \wedge ... \wedge P_Q(v_{i+1}) \wedge v_1 = \text{"0"}$
"where P_Q is $P_\exists$ if $(i+1)$ is odd, or $P_\forall$ if $(i+1)$ is even"

$$\begin{aligned}
&\wedge[PATH_\le(v_1, v_2) \wedge PATH_\le(v_2, v_3) \wedge ... \wedge PATH_\le(v_i, v_{i+1})]\\
&\wedge[\neg\exists v'(PATH_\le(v_1, v') \wedge PATH_\le(v', v_2) \wedge v' \neq v_1 \wedge v' \neq v_2 \wedge P_\forall(v'))]\\
&\quad \wedge[\neg\exists v'(PATH_\le(v_2, v') \wedge PATH_\le(v', v_3) \wedge v' \neq v_2 \wedge v' \neq v_3 \wedge P_\exists(v'))]\\
&\quad \wedge ...\\
&\quad \wedge[\neg\exists v'(PATH_\le(v_i, v') \wedge PATH_\le(v', v_{i+1}) \wedge v' \neq v_i \wedge v' \neq v_{i+1} \wedge P_{Q'}(v'))])_0
\end{aligned}$$

"where $P_{Q'}$ is $P_\forall$ if i is odd, or $P_\exists$ if i is even"

"for $1 \le j \le i$, v_j is the first quantifier of the j-th alternating quantifier block:"
("for $i = k$, v_e is the last element in the block i, which must be |")
$\beta_1 \equiv ({}_0 P_\exists(v_1) \wedge P_\forall(v_2) \wedge P_\exists(v_3) \wedge ... \wedge P_Q(v_k) \wedge P_|(v_e) \wedge v_1 =$"0"
"where P_Q is $P_\exists$ if k is odd, or $P_\forall$ if k is even"

$\wedge [PATH_\le(v_1, v_2) \wedge PATH_\le(v_2, v_3) \wedge ... \wedge PATH_\le(v_k, v_e)]$
$\wedge [\neg\exists v'(PATH_\le(v_1, v') \wedge PATH_\le(v', v_2) \wedge v' \neq v_1 \wedge v' \neq v_2 \wedge P_\forall(v'))]$
$\wedge [\neg\exists v'(PATH_\le(v_2, v') \wedge PATH_\le(v', v_3) \wedge v' \neq v_2 \wedge v' \neq v_3 \wedge P_\exists(v'))]$
$\wedge ...$
$\wedge [\neg\exists v'(PATH_\le(v_k, v') \wedge PATH_\le(v', v_e) \wedge v' \neq v_k \wedge v' \neq v_e \wedge P_{Q'}(v'))])_0$
"where $P_{Q'}$ is $P_\forall$ if k is odd, or $P_\exists$ if k is even"

"L' is a bijection from the X in the i-th alternating quantifier block to V_i,
which preserves $Next_X$ and E_i:" when i is not the index of the last block
$\alpha_{i2} \equiv ({}_0 \forall v'[(PATH_\le(v_i, v') \wedge PATH_\le(v', v_{i+1}) \wedge v' \neq v_{i+1} \wedge P_X(v')) \leftrightarrow \exists y'(L'(v', y'))]$
"domain of L' "

$\wedge \forall y'(V_i(y') \rightarrow \exists v'(L'(v', y')))$ "L' is surjective"
$\wedge \forall s, t, s', t'[[(L'(s,t) \wedge L'(s',t') \wedge s \neq s' \wedge PATH_\le(v_i, s) \wedge PATH_\le(s', v_{i+1}) \wedge PATH_\le(s, s')$
$\wedge (\neg\exists z'(PATH_\le(s, z') \wedge PATH_\le(z', s') \wedge z' \neq s \wedge z' \neq s' \wedge P_X(z')))) \rightarrow E_i(t, t')]$
"L' preseves $Next_X$ in the *ith* alternating block and E_i (implies injectivity)"

$\wedge \forall v' y'[L'(v', y') \rightarrow V_i(y')]$
$\wedge \forall v' y' z'([L'(v', y') \wedge L'(v', z')] \rightarrow (y' = z'))])_0$ "L' is a function"

"L' is a bijection from the X in the i-th alternating quantifier block to V_i,
which preserves $Next_X$ and E_i:"
(when $i = k$)

$\beta_2 \equiv ({}_0 \forall v'[(PATH_\le(v_k, v') \wedge PATH_\le(v', v_e) \wedge P_X(v')) \leftrightarrow \exists y'(L'(v', y'))]$ "domain of L' "
$\wedge \forall y'(V_k(y') \rightarrow \exists v'(L'(v', y')))$ "L' is surjective"
$\wedge \forall s, t, s', t'[[(L'(s,t) \wedge L'(s',t') \wedge s \neq s' \wedge PATH_\le(v_k, s) \wedge PATH_\le(s', v_e) \wedge PATH_\le(s, s')$
$\wedge (\neg\exists z'(PATH_\le(s, z') \wedge PATH_\le(z', s') \wedge z' \neq s \wedge z' \neq s' \wedge P_X(z')))) \rightarrow E_k(t, t')]$
"L' preseves $Next_X$ in the k-*th* alternating block and E_k (implies injectivity)"

$\wedge \forall v' y'[L'(v', y') \rightarrow V_k(y')]$
$\wedge \forall v' y' z'([L'(v', y') \wedge L'(v', z')] \rightarrow (y' = z'))])_0$ "L' is a function"

"For $i = k,\ v_e$ is the last | for the last quantifier in the quantifier prefix":

$\beta_3 \equiv ({}_0\forall v'((SUC_{\leq}(v_e, v') \wedge \neg P_{|}(v')) \wedge P_{|}(v_e)$
$\wedge \forall v'(PATH_{\leq}(v_e, v') \rightarrow (\neg P_{\exists}(v') \wedge \neg P_{\forall}(v')))$ "there are no quantifiers after v_e"

$\wedge \exists z'\, y'\, w' \forall v'[PATH_{\leq}(y', v_e) \wedge ([PATH_{\leq}(v', v_e) \wedge PATH_{\leq}(y', v')] \rightarrow P_{|}(v'))$
"w' is the last quantifier in the block k, z' is the X of the variable quantified"
"y' is the first | for that variable"

$\wedge P_Q(w') \wedge P_X(z') \wedge SUC_{\leq}(z', y') \wedge SUC_{\leq}(w', z')])_0$
"where P_Q is $P_{\exists}$ if k is odd, or P_Q is $P_{\forall}$ if k is even"

(6) $\wedge (\forall x,y,z(V_p(x,y) \wedge V_p(x,z) \rightarrow (y{=}z)$ "V_p is a function"
$\wedge (V_p(x, y) \wedge V_p(z, y) \rightarrow x{=}z)$ "V_p is injective"
$\wedge ((P_X(x) \wedge PRED_{\leq}(x, z) \wedge [P_{\exists}(z) \vee P_{\forall}(z)]) \leftrightarrow \exists y(V_t(y) \wedge V_p(x, y)))$
"domain of V_p","range of V_p"
"V_p is partial bijection:X in quantifier prefix of $\varphi \rightarrow V_t$ in G_t"
$\wedge \forall$s,t,s',t'([V_p(s,s') $\wedge$ V_p(t,t') $\wedge$ E_t(s',t')] $\rightarrow$ ($PATH_{\leq}(s, t)$
$\wedge \forall z'((z' \neq s \wedge z' \neq t \wedge PATH_{\leq}(s, z') \wedge PATH_{\leq}(z', t)) \rightarrow \neg P_X(z'))))$
"V_p preserves $\leq^I$ restricted to the X in the quantifier prefix, and E_t."

(7) $\wedge LINEAR^{2,4}(C, E_C)$

(8) $\wedge LINEAR(ST, E_{ST})$

(9) $\wedge ({}_0\forall s, s', t_1, t_2, k_1, k_2({}_1({}_2({}_3 M(s, t_1, t_2) \wedge M(s, k_1, k_2))_3$
$\rightarrow ({}_3(t_1 = k_1 \wedge t_2 = k_2) \wedge ST(s) \wedge C(t_1, t_2))_3)_2$ "M is a function, $M{:}ST^1 \rightarrow C^2$"

$\wedge ({}_3[M(s,k_1,k_2) \wedge M(t_1,k_1,k_2)] \rightarrow s{=}t_1)_3$ "M is injective"

$\wedge ({}_3 ST(s) \rightarrow \exists t_1', t_2'({}_4 M(s, t_1', t_2'))_4)_3$ "M is total"

$\wedge ({}_3[{}_4 M(s, t_1, t_2) \wedge M(s', k_1, k_2) \wedge PATH_{ST}(s, s')]_4 \rightarrow PATH^{2,4}_{E_C}(t_1, t_2, k_1, k_2))_3)_1)_0$
"M preserves $PATH$ in E_{ST} and E_C"

(10) $\wedge \forall s_1, s_2((C_\wedge(s_1,s_2) \leftrightarrow \neg C_\vee(s_1,s_2)) \wedge (C_\wedge(s_1,s_2) \leftrightarrow \neg C_\neg(s_1,s_2))$
$\wedge (C_\wedge(s_1,s_2) \leftrightarrow \neg C_(\,(s_1,s_2)) \wedge (C_\wedge(s_1,s_2) \leftrightarrow \neg C_)(s_1,s_2))$
$\wedge (C_\wedge(s_1,s_2) \leftrightarrow \neg C_1(s_1,s_2)) \wedge (C_\wedge(s_1,s_2) \leftrightarrow \neg C_0(s_1,s_2)) \wedge ...)$
"$C_\wedge^2, C_\vee^2, C_\neg^2, C_(^2, C_)^2, C_1^2, C_0^2$ are pairwise disjoint"

$\wedge \forall s_1, s_2(C(s_1,s_2) \rightarrow (C_\wedge(s_1,s_2) \vee C_\vee(s_1,s_2) \vee C_\neg(s_1,s_2) \vee C_(\,(s_1,s_2) \vee C_)(s_1,s_2)$
$\vee C_0(s_1,s_2) \vee C_1(s_1,s_2)))$

$\wedge \forall s_1, s_2((C_\wedge(s_1,s_2) \rightarrow C(s_1,s_2)) \wedge (C_\vee(s_1,s_2) \rightarrow C(s_1,s_2)) \wedge (C_\neg(s_1,s_2) \rightarrow C(s_1,s_2))$
$\wedge (C_(\,(s_1,s_2) \rightarrow C(s_1,s_2)) \wedge (C_)(s_1,s_2) \rightarrow C(s_1,s_2)) \wedge (C_0(s_1,s_2) \rightarrow C(s_1,s_2))$
$\wedge (C_1(s_1,s_2) \rightarrow C(s_1,s_2)))$
"$(C_\wedge^2 \bigcup C_\vee^2 \bigcup C_\neg^2 \bigcup C_(^2 \bigcup C_)^2 \bigcup C_1^2 \bigcup C_0^2) = C$"

(11) $\wedge \forall x, y_1, y_2, z_1, z_2 ({}_1({}_2(H_\phi(x,y_1,y_2) \wedge H_\phi(x,z_1,z_2)) \rightarrow ((y_1 = z_1 \wedge y_2 = z_2)$
$\wedge \exists x'(P_(x') \wedge PATH_\leq(x',x)) \wedge C(y_1,y_2)))_2$
"H_ϕ is a function, H_ϕ:quantifier free part of $\varphi \rightarrow C$"

$\wedge ({}_2 H_\phi(x,y_1,y_2) \wedge H_\phi(z,y_1,y_2) \rightarrow x{=}z)_2$ "H_ϕ is injective"

$\wedge \forall y_1', y_2', z_1', z_2', t_1', t_2', v', v_2$
$({}_2({}_3 ST(v') \wedge \neg \exists y(E_{ST}(y,v'))$ "v' is the root in $<ST, E_{ST}>$"

$\wedge E_{ST}(v',v_2) \wedge M(v',y_1',y_2') \wedge M(v_2,z_1',z_2') \wedge E_C(t_1',t_2',z_1',z_2')$
$\wedge PATH_{E_C}(y_1',y_2',y_1,y_2) \wedge PATH_{E_C}(y_1,y_2,t_1',t_2'))_3 \rightarrow \exists x'(H_\phi(x',y_1,y_2)))_2$
"the range of H_ϕ is the first formula in $<C, E_C>$"

$\wedge (\exists x'(P_(x') \wedge PATH_\leq(x',x)) \rightarrow \exists y_1', y_2'(H_\phi(x,y_1',y_2')))_1$
"the domain of H_ϕ is the quantifier free part of φ"

$\wedge \forall x, x', z, y_1, y_2, z_1, z_2[((H_\phi(x,y_1,y_2) \wedge H_\phi(z,z_1,z_2) \wedge E_C(y_1,y_2,z_1,z_2)) \rightarrow$
$(SUC_\leq(x,z) \vee (PATH_\leq(x,z) \wedge \forall x'((PATH_\leq(x,x') \wedge PATH_\leq(x',z)$
$\wedge x' \neq x \wedge x' \neq z) \rightarrow P_|(x'))))$
$\wedge (H_\phi(x,y_1,y_2) \rightarrow ((P_(x) \wedge C_(y_1,y_2)) \vee (P_)(x) \wedge C_)(y_1,y_2))$
$\vee (P_\wedge(x) \wedge C_\wedge(y_1,y_2)) \vee (P_\vee(x) \wedge C_\vee(y_1,y_2))$
$\vee (P_\neg(x) \wedge C_\neg(y_1,y_2)) \vee (P_X(x) \wedge (C_0(y_1,y_2) \vee C_1(y_1,y_2)))))]$
"H_ϕ preserves $\leq_I$ (ignoring |) and E_C, {∧,∨,(,),¬}, and maps X to 0/1"

(12) $\wedge$ "$[(V_1 \bigcap V_2 = \phi) \wedge (V_1 \bigcap V_3 = \phi) \wedge ... \wedge (V_1 \bigcap V_k = \phi)$
$\wedge (V_2 \bigcap V_3 = \phi) \wedge (V_2 \bigcap V_4 = \phi) \wedge ... \wedge (V_2 \bigcap V_k = \phi)$
$\wedge .....$
$\wedge (V_{k-1} \bigcap V_k = \phi)]$"

where "$V_i \bigcap V_j = \phi$" is $\forall x((V_i(x) \rightarrow \neg V_j(x)) \wedge (V_j(x) \rightarrow \neg V_i(x)))$

(13) "$G_2 =< V_2, E_2 >, G_4 =< V_4, E_4 > ... G_{k"} =< V_{k"}, E_{k"} >$ are linear graphs":

(where $k"$ is the index of the last $\forall$ quantifier in the prefix)

$LINEAR(V_2, E_2) \wedge LINEAR(V_4, E_4) \wedge ... \wedge LINEAR(V_{k"}, E_{k"})$

(14)"$B_2 : V_2 \rightarrow \{0,1\} \wedge B_4 : V_4 \rightarrow \{0,1\} \wedge ... \wedge B_{k"} : V_{k"} \rightarrow \{0,1\}$ are total functions"

$\wedge [\forall t, p, p'[B_2(t, p) \wedge B_2(t, p') \rightarrow (p = p')$ "function"
$\wedge (V_2(t) \rightarrow \exists p(B_2(t, p)))$ "total"
$\wedge (B_2(t, p) \rightarrow (p = "0" \vee p = "1"))]$

$\wedge \forall t, p, p'[B_4(t, p) \wedge B_4(t, p') \rightarrow (p = p')$ "function"
$\wedge (V_4(t) \rightarrow \exists p(B_4(t, p)))$ "total"
$\wedge (B_4(t, p) \rightarrow (p = "0" \vee p = "1"))]$
$\wedge .....$
$\wedge \forall t, p, p'[B_{k"}(t, p) \wedge B_{k"}(t, p') \rightarrow (p = p')$ "function"
$\wedge (V_{k"}(t) \rightarrow \exists p(B_{k"}(t, p)))$ "total"
$\wedge (B_{k"}(t, p) \rightarrow (p = "0" \vee p = "1"))]]$

(15) If $k" \neq k,$ then (by $\alpha_{k"1}$ we mean the formula template α_{i1} instantiated with $i = k"$):

$$\bigwedge_{i=2,4,6,...,k"} \exists L'^2 v_1 v_2 v_3 ... v_{k"} v_{k"+1} (\alpha_{k"1} \wedge \alpha_{i2})$$

If $k" = k,$ then (by $\alpha_{k"-2,1}$ we mean the formula template α_{i1} instantiated with $i = k" - 2$):
($k" - 2$ is the previous to the last $\forall$ block, and the subformulas $\beta_1, \beta_2, \beta_3$
take care of the last quantifier block)

$$(\bigwedge_{i=2,4,6,...,k"-2} \exists L'^2 v_1 v_2 v_3 ... v_{k"-1} (\alpha_{k"-2,1} \wedge \alpha_{i2})) \wedge \exists L'^2 v_1 v_2 v_3 ... v_k v_e (\beta_1 \wedge \beta_2 \wedge \beta_3)$$

where the subformulas $\alpha_{i1}, \alpha_{i2}, \beta_1, \beta_2$ and β_3 are the same as in (5) above.

(16) $\wedge \exists \mathbf{U}_1^2 \mathbf{U}_2^2 ... \mathbf{U}_k^2$ ("

$\mathbf{U}_1 : V_1 \to V_t$ – Total injection,

– preserves E_1, E_t,

– includes first node in V_t as U_1(first node in E_1),

$\mathbf{U}_2 : V_2 \to V_t$ – Total injection,

– preserves E_2, E_t,

– includes successor in E_t of node U_1(last node in E_1) in V_t as U_2(first node in E_2),

$\mathbf{U}_3 : V_3 \to V_t$ – Total injection,

– preserves E_3, E_t,

– includes successor in E_t of node U_2(last node in E_2) in V_t as U_3(first node in E_3),

..., and

$\mathbf{U}_k : V_k \to V_t$ – Total injection,

– preserves E_k, E_t,

– includes successor in E_t of node U_{k-1}(last node in E_{k-1}) in V_t as U_k(first node in E_k)")

$\equiv \wedge \exists U_1^2 U_2^2 ... U_k^2 (...$

$\wedge \forall x, y, t, u[((U_k(x,y) \wedge U_k(x,t)) \to y = t)$ "function"

$\wedge (U_k(x,y) \wedge U_k(u,y)) \to x = u)$ "injection"

$\wedge (V_k(x) \to \exists y(U_k(x,y)))$ "total"

$\wedge (U_k(x,y) \to (V_k(x) \wedge V_t(y))$ "$U_k : V_k \to V_t$"

$\wedge ((U_k(x,y) \wedge U_k(u,t) \wedge E_k(x,u)) \to E_t(y,t))$ "preserves E_k"

$\wedge ((U_k(x,y) \wedge U_k(u,t) \wedge E_t(y,t)) \to E_k(x,u))$ "preserves E_t"

$\wedge ((U_{k-1}(x,y) \wedge \neg \exists v(E_{k-1}(x,v)) \wedge E_t(y,t) \wedge \neg \exists v(E_k(v,u) \wedge V_k(u)))$

$\to U_k(u,t))]$)

"includes successor in E_t of node U_{k-1}(last node in E_{k-1}) in V_t as U_k(first node in E_k)"

(17) "B_t *coincides* with $B_1, B_2, ..., B_k$"

$\forall xytpp'[(B_1(t,p) \wedge U_1(t,y) \wedge B_t(y,p')) \to p = p']$

$\wedge$

$\forall xytpp'[(B_2(t,p) \wedge U_2(t,y) \wedge B_t(y,p')) \to p = p']$

$\wedge$

...

$\wedge$

$\forall xytpp'[(B_k(t,p) \wedge U_k(t,y) \wedge B_t(y,p')) \to p = p']$

(18) $\rightarrow [_2 \forall V_o^2 \exists z_0, y_0, z_f, y_f, z_0', y_0', z_f', y_f'$

$[_3 ([_4 [_5 V_o(z_0, y_0) \wedge \neg\exists z', y'(PRED_{\leq}(z_0, z') \wedge V_o(z', y'))$

$\wedge V_o(z_f, y_f) \wedge \neg\exists z', y'(SUC_{\leq}(z_f, z') \wedge V_o(z', y'))$

$\wedge \forall z'([PATH_{\leq}(z_0, z') \wedge PATH_{\leq}(z', z_f)] \rightarrow \exists y'(V_0(z', y')))$

$\wedge \forall y'([PATH_{\leq}(y_0, y') \wedge PATH_{\leq}(y', y_f)] \rightarrow \exists z'(V_0(z', y')))]_5$

"z_0 is the root in *dom*(V_o), z_f is the leaf in *dom*(V_o), y_0 is the root in *ran*(V_o), y_f is the leaf in *ran*(V_o)"

(19) $\wedge \forall x, y, v, w[_5 (V_o(x, y) \rightarrow (P_|(x) \wedge P_|(y)))$

$\wedge ((V_o(x, y) \wedge V_o(x, v)) \rightarrow y = v)$

$\wedge ((V_o(x, y) \wedge V_o(w, y)) \rightarrow x = w)$

$\wedge ((V_o(x, y) \wedge V_o(v, w) \wedge SUC_{\leq}(x, v)) \rightarrow SUC_{\leq}(y, w))]_5$

"V_o is a partial bijection from "|...|" in "QX |...|" to "|...|" in "(...X |...|...)", which also preserves $\leq_I$ "

(20) $\wedge [_5 PRED_{\leq}(z_0, z_0') \wedge P_X(z_0')$

"z_0' is the predecessor of the root in *dom*(V_o), i.e., it is the X in the quantifier prefix"

$\wedge PRED_{\leq}(y_0, y_0') \wedge P_X(y_0')$

"y_0' is the predecessor of the root in *ran*(V_o), i.e., it is the X in the quantifier free part"

$\wedge SUC_{\leq}(z_f, z_f') \wedge \neg P_|(z_f')$ "z_f' is the successor of the leaf in *dom*(V_o)"

$\wedge SUC_{\leq}(y_f, y_f') \wedge \neg P_|(y_f')]_5)_4$ "y_f' is the successor of the leaf in *ran*(V_o)"

(21) $\rightarrow \forall x, x'([_4 (V_p(z_0', x) \wedge B_t(x, x')) \rightarrow (\exists z_1, z_2 (H_\phi(y_0', z_1, z_2)$

$\wedge [(x' = "0" \wedge C_0(z_1, z_2)) \vee (x' = "1" \wedge C_1(z_1, z_2))]))))_4]_3$

"$B_t(V_p(z_0')) = H_\phi(y_0')$"

(22) $\equiv$ "$\forall x$ (the formula in $<C, E_C>$ at stage $x+1$ is the same as the formula at stage x, except for one subformula as in Figure 14)"

$\equiv \forall x[_3 ST(x) \rightarrow \exists E_v^4, f_1, f_2, l_1, l_2, f_1', f_2', l_1', l_2' [_4 \mathbf{(4)} \vee \mathbf{(5)} \vee [_5 \mathbf{(0)} \wedge (_6 \mathbf{(1)} \vee \mathbf{(2)} \vee \mathbf{(3))_6]_5]_4]_3]_2]_1)_0$

The sub formula (4) corresponds to the last transition, i.e. the transition to the last formula in $< C, E_C >$(“0” or “1”). The three sub formulas (1), (2), (3) correspond to the three possible cases (a), (b), (c) as in ***Figure 14***, according to which sort of operation is the one involved in the transition from the formula in stage x to the next formula in $< C, E_C >$.

(0) $({}_6({}_7({}_8\exists y, y_1(E_{ST}(x, y) \wedge E_{ST}(y, y_1))_8$

"x is not the leaf in E_{ST} and is not the predecessor of the leaf "

$$\wedge({}_8\forall s_1, s_2, t_1, t_2, k_1, k_2({}_0({}_1({}_2E_v(s_1, s_2, t_1, t_2) \wedge E_v(s_1, s_2, k_1, k_2))_2$$
$$\rightarrow({}_2(t_1 = k_1 \wedge t_2 = k_2) \wedge C(s_1, s_2) \wedge C(t_1, t_2))_2)_1$$
"E_v is a function, E_v:$C \rightarrow C$ "

$$\wedge({}_2[E_v(s_1, s_2, k_1, k_2) \wedge E_v(t_1, t_2, k_1, k_2)] \rightarrow (s_1 = t_1 \wedge s_2 = t_2))_2)_0)_8)_7$$
"E_v is injective, E_v:C $\rightarrow C$ "

"$\exists$partial injection E_v^4 mapping the formula in $<C, E_C>$ in stage x
to the formula in $<C, E_C>$ in stage $E_{ST}(x)$)"

$$\wedge({}_7(M(x, f_1, f_2) \wedge \text{"}M(E_{ST}(x), E_C(l_1, l_2))\text{"} \wedge \text{"}E_C(l_1, l_2) = (f_1', f_2')\text{"})$$
$$\wedge \text{"}E_C^{-1}(M[E_{ST}(E_{ST}(x))], l_1', l_2')\text{"})$$
"$[(f_1, f_2), (l_1, l_2)]$ and $[(f_1', f_2'), (l_1', l_2')]$
are the delimiters of the two formulas as in Figure 15"

$$\wedge \forall y_1, y_2, z_1, z_2((E_v(y_1, y_2, z_1, z_2)) \rightarrow [((PATH_{E_C}^{2,4}(f_1, f_2, y_1, y_2) \wedge PATH_{E_C}^{2,4}(y_1, y_2, l_1, l_2)))$$
$$\wedge (PATH_{E_C}^{2,4}(f_1', f_2', z_1, z_2) \wedge PATH_{E_C}^{2,4}(z_1, z_2, l_1', l_2'))]$$
$$\wedge E_v(f_1, f_2, f_1', f_2') \wedge E_v(l_1, l_2, l_1', l_2'))_7)_6$$
"E_v maps nodes from the subgraph $[(f_1, f_2), (l_1, l_2)]$ to the subgraph $[(f_1', f_2'), (l_1', l_2')]$"

(1) $({}_{-1}\exists v_1, v_2, w_1, w_2, v_1', v_2', w_1', w_2', p_{11}, p_{12}, p_{21}, p_{22}, p_{31}, p_{32}, p_{11}', p_{12}'$
$({}_0 PATH_{E_C}^{2,4}(f_1, f_2, v_1, v_2) \wedge PATH_{E_C}^{2,4}(w_1, w_2, l_1, l_2) \wedge E_C(p_{11}, p_{12}, p_{21}, p_{22}) \wedge E_C(p_{21}, p_{22}, p_{31}, p_{32})$
$\wedge E_C(v_1, v_2, p_{11}, p_{12}) \wedge E_C(p_{31}, p_{32}, w_1, w_2)$
$\wedge C_(v_1, v_2) \wedge C_)(w_1, w_2) \wedge PATH_{E_C}^{2,4}(f_1', f_2', v_1', v_2') \wedge PATH_{E_C}^{2,4}(w_1', w_2', l_1', l_2')$
$\wedge E_C(v_1', v_2', p_{11}', p_{12}') \wedge E_C(p_{11}', p_{12}', w_1', w_2') \wedge E_v(p_{11}, p_{12}, p_{11}', p_{12}') \wedge E_v(v_1, v_2, v_1', v_2')$
$\wedge E_v(w_1, w_2, w_1', w_2') \wedge C_(v_1', v_2') \wedge C_)(w_1', w_2')$
"$[(v_1, v_2), (w_1, w_2)]$ and $[(v_1', v_2'), (w_1', w_2')]$ define the windows of change (Figure 15 a, b)"

$$\wedge\forall z_{11}, z_{12}, z_{21}, z_{22}, z_{11}', z_{12}', z_{21}', z_{22}'[_1(_2(_3 PATH^{2,4}_{E_C}(f_1, f_2, z_{11}, z_{12}) \wedge PATH^{2,4}_{E_C}(z_{21}, z_{22}, v_1, v_2)$$
$$\wedge E_C(z_{11}, z_{12}, z_{21}, z_{22}) \wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}') \wedge E_v(z_{21}, z_{22}, z_{21}', z_{22}'))_3$$
$$\rightarrow (_3 PATH^{2,4}_{E_C}(f_1', f_2', z_{11}', z_{12}') \wedge PATH^{2,4}_{E_C}(z_{21}', z_{22}', v_1', v_2') \wedge E_C(z_{11}', z_{12}', z_{21}', z_{22}'))_3)_2$$
$$\wedge (_2(_3 PATH^{2,4}_{E_C}(w_1, w_2, z_{11}, z_{12}) \wedge PATH^{2,4}_{E_C}(z_{21}, z_{22}, l_1, l_2) \wedge E_C(z_{11}, z_{12}, z_{21}, z_{22})$$
$$\wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}') \wedge E_v(z_{21}, z_{22}, z_{21}', z_{22}'))_3$$
$$\rightarrow (_3 PATH^{2,4}_{E_C}(w_1', w_2', z_{11}', z_{12}') \wedge PATH^{2,4}_{E_C}(z_{21}', z_{22}', l_1', l_2') \wedge E_C(z_{11}', z_{12}', z_{21}', z_{22}'))_3)_2]_1$$

"E_v preserves E_C outside of the windows, and preserves left and right side of the windows (Figure 15)"

$$\wedge\forall z_{11}, z_{12}, z_{11}', z_{12}'[_1(_2 PATH^{2,4}_{E_C}(f_1, f_2, z_{11}, z_{12}) \wedge PATH^{2,4}_{E_C}(z_{11}, z_{12}, v_1, v_2) \wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}'))_2$$
$$\rightarrow (_2 PATH^{2,4}_{E_C}(f_1', f_2', z_{11}', z_{12}') \wedge PATH^{2,4}_{E_C}(z_{11}', z_{12}', v_1', v_2')$$
$$\wedge [_3(\mathrm{C}_{(}(z_{11}, z_{12}) \wedge \mathrm{C}_{(}(z_{11}', z_{12}')) \vee (\mathrm{C}_{)}(z_{11}, z_{12}) \wedge \mathrm{C}_{)}(z_{11}', z_{12}'))$$
$$\vee (\mathrm{C}_{\wedge}(z_{11}, z_{12}) \wedge \mathrm{C}_{\wedge}(z_{11}', z_{12}')) \vee (\mathrm{C}_{\vee}(z_{11}, z_{12}) \wedge \mathrm{C}_{\vee}(z_{11}', z_{12}'))$$
$$\vee (\mathrm{C}_0(z_{11}, z_{12}) \wedge \mathrm{C}_0(z_{11}', z_{12}')) \vee (\mathrm{C}_1(z_{11}, z_{12}) \wedge \mathrm{C}_1(z_{11}', z_{12}'))$$
$$\vee (\mathrm{C}_{\neg}(z_{11}, z_{12}) \wedge \mathrm{C}_{\neg}(z_{11}', z_{12}'))]_3)_2]_1$$

"E_v preserves symbols in left side of the window"

$$\wedge\forall z_{11}, z_{12}, z_{11}', z_{12}'[_1(_2 PATH^{2,4}_{E_C}(w_1, w_2, z_{11}, z_{12}) \wedge PATH^{2,4}_{E_C}(z_{11}, z_{12}, l_1, l_2) \wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}'))_2$$
$$\rightarrow (_2 PATH^{2,4}_{E_C}(w_1', w_2', z_{11}', z_{12}') \wedge PATH^{2,4}_{E_C}(z_{11}', z_{12}', l_1', l_2')$$
$$\wedge [_3(\mathrm{C}_{(}(z_{11}, z_{12}) \wedge \mathrm{C}_{(}(z_{11}', z_{12}')) \vee (\mathrm{C}_{)}(z_{11}, z_{12}) \wedge \mathrm{C}_{)}(z_{11}', z_{12}'))$$
$$\vee (\mathrm{C}_{\wedge}(z_{11}, z_{12}) \wedge \mathrm{C}_{\wedge}(z_{11}', z_{12}')) \vee (\mathrm{C}_{\vee}(z_{11}, z_{12}) \wedge \mathrm{C}_{\vee}(z_{11}', z_{12}'))$$
$$\vee (\mathrm{C}_0(z_{11}, z_{12}) \wedge \mathrm{C}_0(z_{11}', z_{12}')) \vee (\mathrm{C}_1(z_{11}, z_{12}) \wedge \mathrm{C}_1(z_{11}', z_{12}'))$$
$$\vee (\mathrm{C}_{\neg}(z_{11}, z_{12}) \wedge \mathrm{C}_{\neg}(z_{11}', z_{12}'))]_3)_2]_1$$

"E_v preserves symbols in right side of the window"

$$\wedge [_1(C_0(p_{11}, p_{12}) \wedge C_0(p_{31}, p_{32}) \wedge C_{\wedge}(p_{21}, p_{22}) \wedge C_0(p_{11}', p_{12}'))$$
$$\vee (C_0(p_{11}, p_{12}) \wedge C_0(p_{31}, p_{32}) \wedge C_{\vee}(p_{21}, p_{22}) \wedge C_0(p_{11}', p_{12}'))$$
$$\vee (C_0(p_{11}, p_{12}) \wedge C_1(p_{31}, p_{32}) \wedge C_{\wedge}(p_{21}, p_{22}) \wedge C_0(p_{11}', p_{12}'))$$
$$\vee (C_0(p_{11}, p_{12}) \wedge C_1(p_{31}, p_{32}) \wedge C_{\vee}(p_{21}, p_{22}) \wedge C_1(p_{11}', p_{12}'))$$
$$\vee (C_1(p_{11}, p_{12}) \wedge C_0(p_{31}, p_{32}) \wedge C_{\wedge}(p_{21}, p_{22}) \wedge C_0(p_{11}', p_{12}'))$$
$$\vee (C_1(p_{11}, p_{12}) \wedge C_0(p_{31}, p_{32}) \wedge C_{\vee}(p_{21}, p_{22}) \wedge C_1(p_{11}', p_{12}'))$$
$$\vee (C_1(p_{11}, p_{12}) \wedge C_1(p_{31}, p_{32}) \wedge C_{\wedge}(p_{21}, p_{22}) \wedge C_1(p_{11}', p_{12}'))$$
$$\vee (C_1(p_{11}, p_{12}) \wedge C_1(p_{31}, p_{32}) \wedge C_{\vee}(p_{21}, p_{22}) \wedge C_1(p_{11}', p_{12}'))]_1)_0)_{-1}$$

"In (p_{11}', p_{12}') we get the result of applying the operator θ in (p_{21}, p_{22}) to the Boolean values b_1, in (p_{11}, p_{12}), and b_2, in (p_{31}, p_{32}) (Figure 14a)"

(2) $({}_{-1}\exists v_1, v_2, w_1, w_2, v_1', v_2', w_1', w_2', p_{11}, p_{12}, p_{21}, p_{22}, p_{11}', p_{12}'$

$({}_0 PATH_{E_C}^{2,4}(f_1, f_2, v_1, v_2) \wedge PATH_{E_C}^{2,4}(w_1, w_2, l_1, l_2) \wedge E_C(p_{11}, p_{12}, p_{21}, p_{22})$

$\wedge E_C(v_1, v_2, p_{11}, p_{12}) \wedge E_C(p_{21}, p_{22}, w_1, w_2)$

$\wedge C_(v_1, v_2) \wedge C_)(w_1, w_2) \wedge PATH_{E_C}^{2,4}(f_1', f_2', v_1', v_2') \wedge PATH_{E_C}^{2,4}(w_1', w_2', l_1', l_2')$

$\wedge E_C(v_1', v_2', p_{11}', p_{12}') \wedge E_C(p_{11}', p_{12}', w_1', w_2') \wedge E_v(p_{11}, p_{12}, p_{11}', p_{12}')$

$\wedge E_v(v_1, v_2, v_1', v_2') \wedge E_v(w_1, w_2, w_1', w_2') \wedge C_(v_1', v_2') \wedge C_)(w_1', w_2')$

"$[(v_1, v_2), (w_1, w_2)]$ and $[(v_1', v_2'), (w_1', w_2')]$ define the windows of change (Figure 15 a, b)"

$\wedge \forall z_{11}, z_{12}, z_{21}, z_{22}, z_{11}', z_{12}', z_{21}', z_{22}' [_1({}_2({}_3 PATH_{E_C}^{2,4}(f_1, f_2, z_{11}, z_{12}) \wedge PATH_{E_C}^{2,4}(z_{21}, z_{22}, v_1, v_2)$

$\wedge E_C(z_{11}, z_{12}, z_{21}, z_{22}) \wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}') \wedge E_v(z_{21}, z_{22}, z_{21}', z_{22}'))_3$

$\rightarrow ({}_3 PATH_{E_C}^{2,4}(f_1', f_2', z_{11}', z_{12}') \wedge PATH_{E_C}^{2,4}(z_{21}', z_{22}', v_1', v_2') \wedge E_C(z_{11}', z_{12}', z_{21}', z_{22}'))_3)_2$

$\wedge ({}_2({}_3 PATH_{E_C}^{2,4}(w_1, w_2, z_{11}, z_{12}) \wedge PATH_{E_C}^{2,4}(z_{21}, z_{22}, l_1, l_2) \wedge E_C(z_{11}, z_{12}, z_{21}, z_{22})$

$\wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}') \wedge E_v(z_{21}, z_{22}, z_{21}', z_{22}'))_3$

$\rightarrow ({}_3 PATH_{E_C}^{2,4}(w_1', w_2', z_{11}', z_{12}') \wedge PATH_{E_C}^{2,4}(z_{21}', z_{22}', l_1', l_2') \wedge E_C(z_{11}', z_{12}', z_{21}', z_{22}'))_3)_2]_1$

"E_v preserves E_C outside of the windows, and preserves left and right side of the windows (Figure 15)"

$\wedge \forall z_{11}, z_{12}, z_{11}', z_{12}' [_1({}_2 PATH_{E_C}^{2,4}(f_1, f_2, z_{11}, z_{12}) \wedge PATH_{E_C}^{2,4}(z_{11}, z_{12}, v_1, v_2) \wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}'))_2$

$\rightarrow ({}_2 PATH_{E_C}^{2,4}(f_1', f_2', z_{11}', z_{12}') \wedge PATH_{E_C}^{2,4}(z_{11}', z_{12}', v_1', v_2')$

$\wedge [_3(\mathrm{C}_(z_{11}, z_{12}) \wedge \mathrm{C}_(z_{11}', z_{12}')) \vee (\mathrm{C}_)(z_{11}, z_{12}) \wedge \mathrm{C}_)(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_\wedge(z_{11}, z_{12}) \wedge \mathrm{C}_\wedge(z_{11}', z_{12}')) \vee (\mathrm{C}_\vee(z_{11}, z_{12}) \wedge \mathrm{C}_\vee(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_0(z_{11}, z_{12}) \wedge \mathrm{C}_0(z_{11}', z_{12}')) \vee (\mathrm{C}_1(z_{11}, z_{12}) \wedge \mathrm{C}_1(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_\neg(z_{11}, z_{12}) \wedge \mathrm{C}_\neg(z_{11}', z_{12}'))]_3)_2]_1$

"E_v preserves symbols in left side of the window"

$\wedge \forall z_{11}, z_{12}, z_{11}', z_{12}' [_1({}_2 PATH_{E_C}^{2,4}(w_1, w_2, z_{11}, z_{12}) \wedge PATH_{E_C}^{2,4}(z_{11}, z_{12}, l_1, l_2) \wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}'))_2$

$\rightarrow ({}_2 PATH_{E_C}^{2,4}(w_1', w_2', z_{11}', z_{12}') \wedge PATH_{E_C}^{2,4}(z_{11}', z_{12}', l_1', l_2')$

$\wedge [_3(\mathrm{C}_(z_{11}, z_{12}) \wedge \mathrm{C}_(z_{11}', z_{12}')) \vee (\mathrm{C}_)(z_{11}, z_{12}) \wedge \mathrm{C}_)(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_\wedge(z_{11}, z_{12}) \wedge \mathrm{C}_\wedge(z_{11}', z_{12}')) \vee (\mathrm{C}_\vee(z_{11}, z_{12}) \wedge \mathrm{C}_\vee(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_0(z_{11}, z_{12}) \wedge \mathrm{C}_0(z_{11}', z_{12}')) \vee (\mathrm{C}_1(z_{11}, z_{12}) \wedge \mathrm{C}_1(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_\neg(z_{11}, z_{12}) \wedge \mathrm{C}_\neg(z_{11}', z_{12}'))]_3)_2]_1$

"E_v preserves symbols in right side of the window"

$\wedge [_1(C_0(p_{21}, p_{22}) \wedge C_\neg(p_{11}, p_{12}) \wedge C_1(p_{11}', p_{12}'))$

$\vee (C_1(p_{21}, p_{22}) \wedge C_\neg(p_{11}, p_{12}) \wedge C_0(p_{11}', p_{12}'))]_1)_0)_{-1}$

"In (p_{11}', p_{12}') we get the result of applying $\neg$, in (p_{11}, p_{12}), to the Boolean value b_1, in (p_{21}, p_{22}) (Figure 14b)"

(3) $({}_{-1}\exists v_1, v_2, w_1, w_2, p_{11}, p_{12}, p_{11}', p_{12}'$

$({}_0 PATH^{2,4}_{E_C}(f_1, f_2, v_1, v_2) \wedge PATH^{2,4}_{E_C}(w_1, w_2, l_1, l_2) \wedge E_C(v_1, v_2, p_{11}, p_{12}) \wedge E_C(p_{11}, p_{12}, w_1, w_2)$

$\wedge C_{(}(v_1, v_2) \wedge C_{)}(w_1, w_2) \wedge PATH^{2,4}_{E_C}(f_1', f_2', p_{11}', p_{12}') \wedge PATH^{2,4}_{E_C}(p_{11}', p_{12}', l_1', l_2')$

$\wedge E_v(p_{11}, p_{12}, p_{11}', p_{12}')$

$[(v_1, v_2), (w_1, w_2)]$ and $[(p_{11}', p_{12}'), (p_{11}', p_{12}')]$ define the windows of change (Figure 15c)"

$\wedge \forall z_{11}, z_{12}, z_{21}, z_{22}, z_{11}', z_{12}', z_{21}', z_{22}' [_1({}_2({}_3 PATH^{2,4}_{E_C}(f_1, f_2, z_{11}, z_{12}) \wedge PATH^{2,4}_{E_C}(z_{21}, z_{22}, v_1, v_2)$

$\wedge E_C(z_{11}, z_{12}, z_{21}, z_{22}) \wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}') \wedge E_v(z_{21}, z_{22}, z_{21}', z_{22}'))_3$

$\rightarrow ({}_3 PATH^{2,4}_{E_C}(f_1', f_2', z_{11}', z_{12}') \wedge E_C(z_{11}', z_{12}', z_{21}', z_{22}') \wedge PATH^{2,4}_{E_C}(z_{21}', z_{22}', p_{11}', p_{12}'))_3)_2$

$\wedge ({}_2({}_3 PATH^{2,4}_{E_C}(w_1, w_2, z_{11}, z_{12}) \wedge PATH^{2,4}_{E_C}(z_{21}, z_{22}, l_1, l_2) \wedge E_C(z_{11}, z_{12}, z_{21}, z_{22})$

$\wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}') \wedge E_v(z_{21}, z_{22}, z_{21}', z_{22}'))_3$

$\rightarrow ({}_3 PATH^{2,4}_{E_C}(z_{21}', z_{22}', l_1', l_2') \wedge E_C(z_{11}', z_{12}', z_{21}', z_{22}') \wedge PATH^{2,4}_{E_C}(p_{11}', p_{12}', z_{11}', z_{12}'))_3)_2]_1$

"E_v preserves E_C outside of the windows, and preserves left and right side of the windows (Figure 15)"

$\wedge \forall z_{11}, z_{12}, z_{11}', z_{12}' [_1({}_2 PATH^{2,4}_{E_C}(f_1, f_2, z_{11}, z_{12}) \wedge PATH^{2,4}_{E_C}(z_{11}, z_{12}, v_1, v_2) \wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}'))_2$

$\rightarrow ({}_2 PATH^{2,4}_{E_C}(f_1', f_2', z_{11}', z_{12}') \wedge PATH^{2,4}_{E_C}(z_{21}', z_{22}', p_{11}', p_{12}')$

$\wedge [_3(\mathrm{C}_{(}(z_{11}, z_{12}) \wedge \mathrm{C}_{(}(z_{11}', z_{12}')) \vee (\mathrm{C}_{)}(z_{11}, z_{12}) \wedge \mathrm{C}_{)}(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_{\wedge}(z_{11}, z_{12}) \wedge \mathrm{C}_{\wedge}(z_{11}', z_{12}')) \vee (\mathrm{C}_{\vee}(z_{11}, z_{12}) \wedge \mathrm{C}_{\vee}(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_0(z_{11}, z_{12}) \wedge \mathrm{C}_0(z_{11}', z_{12}')) \vee (\mathrm{C}_1(z_{11}, z_{12}) \wedge \mathrm{C}_1(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_{\neg}(z_{11}, z_{12}) \wedge \mathrm{C}_{\neg}(z_{11}', z_{12}'))]_3)_2]_1$

"E_v preserves symbols in left side of the window"

$\wedge \forall z_{11}, z_{12}, z_{11}', z_{12}' [_1({}_2 PATH^{2,4}_{E_C}(w_1, w_2, z_{11}, z_{12}) \wedge PATH^{2,4}_{E_C}(z_{11}, z_{12}, l_1, l_2) \wedge E_v(z_{11}, z_{12}, z_{11}', z_{12}'))_2$

$\rightarrow ({}_2 PATH^{2,4}_{E_C}(p_{11}', p_{12}', z_{11}', z_{12}') \wedge PATH^{2,4}_{E_C}(z_{11}', z_{12}', l_1', l_2')$

$\wedge [_3(\mathrm{C}_{(}(z_{11}, z_{12}) \wedge \mathrm{C}_{(}(z_{11}', z_{12}')) \vee (\mathrm{C}_{)}(z_{11}, z_{12}) \wedge \mathrm{C}_{)}(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_{\wedge}(z_{11}, z_{12}) \wedge \mathrm{C}_{\wedge}(z_{11}', z_{12}')) \vee (\mathrm{C}_{\vee}(z_{11}, z_{12}) \wedge \mathrm{C}_{\vee}(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_0(z_{11}, z_{12}) \wedge \mathrm{C}_0(z_{11}', z_{12}')) \vee (\mathrm{C}_1(z_{11}, z_{12}) \wedge \mathrm{C}_1(z_{11}', z_{12}'))$

$\vee (\mathrm{C}_{\neg}(z_{11}, z_{12}) \wedge \mathrm{C}_{\neg}(z_{11}', z_{12}'))]_3)_2]_1$

"E_v preserves symbols in right side of the window"

$\wedge [_1(C_0(p_{11}, p_{12}) \wedge C_0(p_{11}', p_{12}')) \vee (C_1(p_{11}, p_{12}) \wedge C_1(p_{11}', p_{12}'))]_1)_0)_{-1}$

"In (p_{11}', p_{12}') we get a copy of the Boolean value b_1, in (p_{11}, p_{12}) (Figure 14c)"

(4) $({}_0 \exists y({}_1 E_{ST}(x, y) \wedge \neg \exists z(E_{ST}(y, z))$

$\wedge \exists p_{11}, p_{12}, p_{11}', p_{12}' ({}_2 M(x, f_1, f_2) \wedge M(y, p_{11}', p_{12}') \wedge E_C(f_1, f_2, p_{11}, p_{12})$

$\wedge E_C(p_{11}, p_{12}, l_1, l_2) \wedge E_C(l_1, l_2, p_{11}', p_{12}')$

$\wedge \neg \exists p_{21}', p_{22}' (E_C(p_{11}', p_{12}', p_{21}', p_{22}'))$

$\wedge C_{(}(f_1, f_2) \wedge C_{)}(l_1, l_2) \wedge [_3(C_1(p_{11}, p_{12}) \wedge C_1(p_{11}', p_{12}'))$

$\vee (C_0(p_{11}, p_{12}) \wedge C_0(p_{11}', p_{12}'))]_3)_2)_1)_0$

"x is the predecessor of the leaf in E_{ST}, so that this is the last transition" (See Figure 16)
Note that the last transition is necessarily an instance of case c in Figure 14.

$$\textbf{(5)}(_0\ \neg\exists y(E_{ST}(x,y)) \qquad \text{"}x\text{ is the leaf in }E_{ST}\text{"}$$
$$\wedge\exists p_1', p_2'(M(x,p_1',p_2')\wedge\neg\exists y_1', y_2'(E_C(p_1',p_2',y_1',y_2')) \quad \text{"}(p_1', p_2')\text{ is a leaf in }E_C\text{"}$$
$$\wedge C_1(p_1', p_2'))))_0$$
$$\text{"the last formula in } <C, E_C> \text{ is "1" "}$$

5.3 Expressing SATQBF in Third-Order Logic

In this section we give a top down presentation of the sketch of the TO formula for *SATQBF* . We do not include the details of the sub-formulas whose expression in TO is straightforward, taking into account the detailed exposition of the SO formula for $SATQBF_k$ in section 5.2.

Roughly, we first express the existence of an alternating valuation suitable for a given *QBF* formula φ . And then we proceed to evaluate that alternating valuation by evaluating each leaf valuation on a propositional formula ϕ on {0, 1} which we build from φ and the leaf valuation. For this last part we use the sub formulas that we have built for $SATQBF_k$.

Input formula φ :

$$\varphi = \exists x_{11}...\exists x_{1l_1}\forall x_{21}...\forall x_{2l_2}\exists x_{31}...\exists x_{3l_3}.....Qx_{k1}...Qx_{kl_k}(\varphi'(x_{11}...x_{1l_1}, x_{21}...x_{2l_2}, x_{31}...x_{3l_3},..., x_{k1}...x_{kl_k}))$$

Where k is **arbitrary**, and the formula φ' is the quantifier free part of φ . We use the same encoding as in $SATQBF_k$.

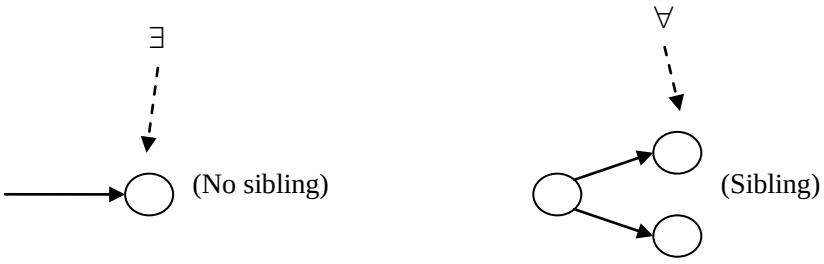


Figure 17

As before, we will also use four levels of abstraction to build a sketch for a TO formula to express *SATQBF*.

1st Level:

"There is an alternating valuation a_v suitable for φ, which satisfies φ"

2nd Level:

$$\exists V_T^{3,1}, E_T^{3,2}, B_T^{3,2} : V_T \to \{0,1\}, V_t^{2,1}, E_t^{2,2}$$

(2.1) ("$G_t = (V_t, E_t)$ is a linear graph which represents the sequence of quantified variables in φ"

$\wedge$

(2.2) "(V_T, E_T) is a binary tree with all its leaves at the same depth, which is in turn equal to the length of (V_t, E_t)"

$\wedge$

(2.3) "(V_T, E_T, B_T) is an **alternating valuation** a_v **suitable** for φ, i.e., all the nodes in (V_T, E_T) whose depth correspond to a universally quantified variable in

the quantifier prefix of φ have exactly 1 sibling, and its value under B_T is different than that of the given node, and all the nodes whose depth correspond to an existentially quantified variable in the quantifier prefix of φ are either the root or have no siblings" (See Figure 17)

$\wedge$

(2.4) "Every **leaf valuation** in (V_T, E_T, B_T) satisfies φ'")

3rd Level:

$$\exists V_T^{3,1}, E_T^{3,2}, B_T^{3,2}, V_t^{2,1}, E_t^{2,2}, V_p^{2,2}($$

Expressing (2.1):

$[_1$"G_t=<V_t^1, E_t^2> is a linear graph" $\equiv LINEAR(V_t, E_t)$ **(1)**

$\wedge$"The length of G_t is equal to the number of variables in the quantifier prefix of φ"

$\equiv [_2$"V_p is a partial bijection from the quantifier prefix of φ (restricted to the X in the quantifier prefix) to V_t, which maps every X in φ to its corresponding node in G_t, and which preserves E_t and $\leq^I$ in G_t, φ (restricted to the X in the quantifier prefix)."$]_2]_1$ **(2)**

In 4th level (subformula 2): See (1) in 5.2.1.2.1 in SATQBF$_k$.

Expressing (2.2):

$\wedge[_1$"(V_T, E_T) is a connected graph that has one root and one or more leaves." **(1)**

$\wedge$"in (V_T, E_T) except for the root, all nodes have id=1" **(2)**

$\wedge$"in (V_T, E_T) except for the leaves, all nodes have od=1 or 2" **(3)**

$\wedge$"in (V_T, E_T) all leaves have the same depth, which is in turn equal to the length of (V_t, E_t)"

$\equiv[_2$"$\forall X^{2,1}(_3(X$ is a leaf in $(V_T, E_T)) \rightarrow (_4 \exists S_D^{'3,1} \subseteq V_T, \exists$ bijection $D^{'3,2}$ **(4)**
$: V_t$ to $S_D^{'}$ which preserves E_t and $E_T|_{S_D^{'}}$"

$\wedge S_D^{'}(X)$ **(5)**

$\wedge$"$D^{'-1}(X)$ is the leaf in $G_t = (V_t, E_t)$" **(6)**

$\wedge$"$S_D^{'}$ includes the root of (V_T, E_T)")$_4)_3]_2]_1$ **(7)**

In 4th level (subformulas 1, 2, 3):

$\exists R^1[\forall Z^1(V_T(Z) \rightarrow PATH_{E_T}^3(R, Z)) \wedge \neg \exists S_1^1(E_T(S_1, R))$

$\wedge \exists S_1^1(\neg \exists S_2^1(E_T(S_1, S_2)))$

$\wedge \forall Z^1((V_T(Z) \wedge Z \neq R) \rightarrow \exists S_1^1(E_T(S_1, Z) \wedge \forall S_2^1(E_T(S_2, Z) \rightarrow S_1 = S_2)))]$

$\wedge \forall Z^1(V_T(Z) \rightarrow \neg \exists S_1^1 S_2^1 S_3^1(S_1 \neq S_2 \wedge S_2 \neq S_3 \wedge S_1 \neq S_3 \wedge E_T(Z, S_1) \wedge E_T(Z, S_2) \wedge E_T(Z, S_3)))$

Expressing (2.3):

See Figures 8, 10 in $SATQBF_k$ for G_t, V_p.

$\wedge[_1$"B_T is a total function from V_T to {0,1}" **(1)**

$\wedge[_2$"$\forall x \in V_t$, $\forall S_D^{3,1} \subseteq V_T$ s.t. $(S_D, E_T|_{S_D})$ is a linear graph which includes
the root of (V_T, E_T) **(2)**

$\forall$ bijection $D^{3,2}$ from the initial subgraph of G_t up to x, to S_D s.t. D preserves E_t and $E_T|_{S_D}$"

$(_3[_4(_5$"the predecessor of $V_p^{-1}(x)$ in E^I is "$\forall$" $")_5$ **(3)**

$\rightarrow (_5$"$D(x)$ has exactly one sibling in (V_T, E_T),

and B_T of that sibling is $\neq$ than $B_T(D(x))")_5]_4$ **(4)**

$\wedge[_4(_5$"the predecessor of $V_p^{-1}(x)$ in E^I is "$\exists$" $")_5$ **(5)**

$\rightarrow (_5$"$D(x)$ has no sibling in (V_T, E_T), or $D(x)$ is the root in $(V_T, E_T)")_5]_4)_3]_2]_1$ **(6)**

Expressing (2.4):

$\wedge[_1$"$\forall S_V^{3,1} \subseteq V_T$ s.t. $(S_V, E_T|_{S_V})$ is a linear graph which includes the root and a leaf of (V_T, E_T)" **(1)**

$(_2$"$\exists$bijection $D^{3,2}: V_t \rightarrow S_V$, which preserves E_t and $E_T|_{S_V}$" **(2)**

$(_3$"$\exists B_t: V_t \rightarrow \{0,1\}$, which coincides with $B_T(S_V)$ w.r.t. D" **(3)**

$\wedge$"the leaf valuation represented by (V_t, E_t, B_t) satisfies the quantifier free subformula φ' of $\varphi")_3)_2]_1)_0$ **(4)**

In 4th level (subformula 4): See 5.2.1.2.2 in SATQBF$_k$.

6 Conclusion and Future Work

In this thesis, we have given a very brief introduction to Finite Model Theory and relational databases, the relationship between logic languages and complexity classes, the basic notions of First-Order logic, Second-Order logic, Third-Order logic and a few related examples. It gives us an overall picture on what logic languages are, how they relate to complexity classes and why they are so important.

The major purposes of this thesis are the following two points. Firstly, a non trivial Second-Order logic formula is built to express a non trivial property, namely the problem known as $SATQBF_k$, which is defined, for some $k>0$ as the class of quantified propositional formulas with k alternated blocks of quantifiers, starting with an existential block, which are true. As we know that the expressibility of SO (Second-Order logic) equals the class PH (Polynomial-time hierarchy), and that for every $k \geq 0$, $SATQBF_k$ is a complete problem for the level Σ_k^P of the PH, there must be a formula in SO which expresses $SATQBF_k$. We give a detailed explanation on the use of SO to express $SATQBF_k$. Secondly, using the knowledge gained from the first point, we give a top down presentation of the sketch of a Third Order Logic formula for $SATQBF$, which is defined similarly but where the number of blocks, k, is arbitrary. This is a problem known to be complete for PSPACE, which means that the existence of a Second-Order logic formula is very unlikely, since it would imply that PSPACE = PH.

Tentatively, I will continue my research on the theoretical parts of Higher Order logics. Maybe I will bring some application perspectives into it such as the construction of a query language for relational databases based on Higher-Order Logics. Also, and following my master thesis, it is also worth investigating, as a part of my PHD work, how the strategy used here to define $SATQBF_k$ can be adapted to define $AUTOSAT$ ([MP96], [Fer08]) for different fragments of Higher-Order logics. We know that for every k there is a reduction from $SATQBF_k$ to $AUTOSAT(\sum_k^0)$. $AUTOSAT(\sum_k^0)$ is the class of first order formulas with up to k alternations of blocks of quantifiers which represented as finite structures satisfy themselves. In fact, same as $SATQBF_k$, $AUTOSAT(\sum_k^0)$ is complete for the level $\sum_k^p$ of the PH (Polynomial Hierarchy). It could be the case that this approach leads to tighter upper bounds than those in the work reported in those articles.

7 References

[CH80] Chandra A., Harel D.. *Computable Queries for Relational Data Bases.* Journal of Computer and System Sciences 21(2), pp. 156-178, 1980.

[Cod72] Codd E. F. (1972). Relational completeness of database sublanguage. In *Data Base Systems (*Rustin, Ed.). Prentice Hall. Pp. 65-98, 1972.

[EF99] Heinz-Dieter Ebbinghaus and J¨org Flum. *Finite Model Theory*. Perspectives in Mathematical Logic. Springer, Berlin Heidelberg New York, 2nd edition, 1999.

[Fer08] Ferrarotti, F. A., *Expressibility of Higher-Order Logics on Relational Databases: Proper Hierarchies*, PhD thesis, Massey University, 2008.

[FT04] Ferrarotti, F.A., Turull-Torres, J.M., *Using Higher Order Quantification in Logical Query languages.* Proceedings of 3rd Chilean Database Workshops, Arica, Chile, 2004.

[GJ03] Garey, M.R., Johnson, D.S., Computers and Intractability: A Guide to the Theory of NP-Completeness, Bell Laboratories, New Jersey, 2003.

[HT06] Hella, L., Turull-Torres, J.M., Computing queries with higher order logics. Theor. Comput. Sci.355, 197-214 (2006).

[Imm99] Immerman, N., Descriptive complexity, Springer-Verlag, New York, 1999.

[Lib04] Libkin, L., Elements of Finite Model Theory, Springer-Verlag, Berlin, 2004.

[MP96] Johann A. Makowsky and Yachin B. Pnueli. Arity and alternation in second-order logic. *Ann. Pure Appl. Logic*, 78(1-3):189–202, 1996.

[RG03] Ramakrishnan R. & Gehrke J.. (2003). *Database Management Systems* (3rd Edition). New York: McGraw-Hill Companies, Inc.

[RT08] Ren, W., Turull-Torres, J.M., *Cubic graph in Second Order Logic. Massey University 2008.*

[Sto76] Stockmeyer, L.J., The polynomial-time hierarchy. *Theor. Comput. Sci.*, 3(1):1–22, 1976.

[Sto87] Stockmeyer, L.J., Classifying the Computational Complexity of Problems. Journal of Symbolic Logic, Volume 52, Issue 1 (Mar., 1987), 1-43.

[TT03] Turull-Torres J. M. (2003). *Notes on Relational Calculus*. Information Science Research Centre, Massey University.

[Var82] Vardi M. Y. (1982). *The Complexity of Relational Query Languages.* Association for Computing Machinery. 5: 137-146.