Reachability Analysis In Graph Transformation Systems

Technical Report

by

Abhisekh Sankaran

under the guidance of

Prof. Supratik Chakraborty

Computer Science & Engineering Department
Indian Institute of Technology
Bombay
2007
## Contents

1 Introduction ..................................................... 1  
   1.1 Graph Transforming Systems ................................. 3  
   1.2 Earlier Work On Graph Transformation Systems .............. 4  

2 The Library System ............................................. 6  
   2.1 State of the library system .................................. 8  
   2.2 Operations of the Library System .............................. 8  
   2.3 Initial State Conditions and Safety Properties .......... 12  

3 Undecidability of Reachability in Graph Transforming Systems 14  

4 Reachability Analysis Using Induction .......................... 17  
   4.1 InductionSolve .............................................. 18  
   4.2 Intuitive Idea behind InductionSolve ......................... 18  
   4.3 Proof Of Correctness ....................................... 19  

5 Logical Formalisation of Graph Transformation Systems ........ 23  
   5.1 Preliminaries from Mathematical Logic ....................... 23  
   5.2 Defining a State ............................................ 25  
   5.3 Defining A Graph Transformation System $S$ .................. 26  
   5.4 Transition Relation on States Defined by $S$ ................. 30  

6 Graph Transformation Systems With Special Sub-actions .......... 32  

7 A Special Kind of Graph Transformation Systems ............... 37  

8 Bounded Model Property ........................................ 43
9 Graph Transformation Systems Having Bounded Models 48

9.1 Class 1 ................................................................. 48
   9.1.1 An Example - The Library System ......................... 48
   9.1.2 Generalization .................................................. 56

9.2 Class 2 ................................................................. 61
   9.2.1 An Example ....................................................... 61
   9.2.2 Generalisation ................................................... 65

10 Deciding The Formulae of InductionSolve 71

10.1 Special GTSes, Safety Properties and Initial State Conditions .......... 71
10.2 Bounded Models For IS1(k) and IS2(k) ................................... 73
10.3 Decision Procedure For InductionSolve .................................... 78
10.4 Checking the Library System .............................................. 81

11 Some studies on the EBS property 83

11.1 A Syntactic Subclass of EBS ......................................... 83
11.2 EBS and semi-linear sets ............................................... 89

12 Conclusion and Future Work 94
Chapter 1

Introduction

State transition systems are a useful mathematical model as they can be used to express the behaviour of wide variety of systems. Often it is desired to check properties that must be true at any point of evolution of the system. Such properties are called safety properties. The checking of these properties in the system can be translated to a reachability analysis in the state-transition model of the system, where the reachability of states of the system violating the property is checked. Thus reachability analysis plays a central role in the verification of the behaviour of systems.

For finite state systems, a variety of techniques have been developed to deal with the problem of reachability. Usually these systems are hardware systems though there are many software systems which are finite state. A simple example of a finite state system is shown below.
It consists of a clocked flip-flop whose output ($Q$) is XOR-ed with an external input $X$ to form the input ($D$) of the flip-flop. The delay of the XOR gate is negligible and that of the flip-flop is about the same as the width of the clock. A state is defined by variables $D$ and $Q$. The 4 states of the circuit are shown in the state-transition graph above. Input $X$ causes transitions between states. An unlabeled edge means that the Clk input itself causes the transition. In this graph, the question of reachability though is a trivial one; every state is reachable from every other.

Reachability in finite state systems is known to be decidable. Explicit representation techniques which operate on the state-transition graph of the system, include depth-first searches (DFS) or breadth-first searches (BFS) for the property violating states, CTL model checking on explicit representation of Kripke structures and LTL model checking by Tableau construction [WOL87]. However several realistic systems have states whose number runs into billions and hence explicit representation of the state-transition graph becomes impractical. Such systems are implicitly represented by their initial states and a transition relation which tells the next states of the system given any present state. BDD-based techniques like symbolic model checking [KLM92] take this implicit representation and use characteristic functions to represent sets of states and transition relations and do symbolic DFS or BFS or symbolic fixpoint computations. There are tools for performing automated reachability analysis of systems. Two of these are SPIN [MP86] and NuSMV [CCG02]. There are tools which do reachability analysis in systems where timing is involved. Some of these include Kronos [DOTY96], Uppaal, STeP [MS94] and Hytech. Recently SAT-based techniques [BCC99] are becoming very popular due to the rapid progress that SAT solving has seen in recent times. Since the first development of the basic search based algorithm proposed by Davis, Putnam, Logemann and Loveland [DPLL62] about 40 years ago, great progress has been made in SAT-solving owing to the active research and more recently, very clever heuristics. SATO, Chaff, Grasp, Berkmin, satz, cnfs are some of the extremely fast solvers that handle millions of clauses with several hundreds of thousands of variables. There are also inductive techniques that use inductive reasoning in doing reachability analysis [MSG00].

Reachability checking in infinite state systems however is in general undecidable. Such systems come primarily from the program verification domain. A simple example of an infinite state system is the following program.

\[
\begin{align*}
i &:= 0; \textbf{while}(1) \ i := i + 1;
\end{align*}
\]

where the state variable is $i$. A simple property to check is whether there is a state in which $6i^2 - 7i = 5$ is true. (Clearly there isn’t).

Reachability analysis in infinite state systems essentially use abstraction techniques [CGJ03] in which the infinite state space of the system is partitioned into a finite number of classes. These are then regarded as abstract states of an abstract state-transition graph which now being finite, allows the use of the wide array of techniques for finite state reachability analysis. The abstract state-transition graph is an overapproximation of the concrete state-transition graph of the system so that if a bad abstract state is not reachable, then surely a
bad concrete state is also not reachable. But a path to a bad abstract state from an abstract initial state might be actually spurious when the path is mapped back to the concrete state-transition graph, in which case refinements to the abstraction are done. Predicate abstraction[20] is a very important abstraction technique in which states of the concrete state space are grouped using predicates. It combines theorem proving and model checking techniques. Control and Data[31] abstraction are also important abstraction techniques in verification. While reachability in infinite state systems is undecidable in general, it is decidable for certain infinite state systems. Location reachability in push down automata is one such example. It is thus possible to make the problem of reachability decidable, by exploiting the specific properties and structure of the transition relation of special classes of infinite systems.

1.1 Graph Transforming Systems

In this report, we consider infinite state transition systems whose states are themselves graphs(possibly infinite). The graphs we consider are typed graphs, i.e. the nodes and edges have types associated with them. A transition from one state to another involves a graph transformation i.e. addition and deletion of nodes and edges as well as change of types associated with the nodes and edges. The transition from one state to another is caused by a finite number of operations. Each operation takes in a set of parameters and is expressed as a finite set of guard-action pairs. Each guard checks if a certain condition is true in a state and the corresponding action describes the graph transformation for that guard-action pair. The action is taken only if the guard evaluates to true. If the guard evaluates to false, then no-action is taken on any node or edge. For any operation, given a set of parameters, the next state is obtained by looking at all those guards which become true in the current state and performing cumulatively, the graph transformations described by each of the actions for which the corresponding guard became true. In general, there is non-determinism in the actions so that an operation, for a given set of parameters, can produce multiple next states. For a state, the set of all next states is then obtained by considering all operations and for each operation, all possible parameter values for it and for each parameter value, the set of all next states for that operation and those parameter values. The Graph Transformation System is then described as the set of all operations. The system starts from a set of initial states i.e. a set of initial graphs. We are now interested in knowing whether certain target graphs having certain properties are reachable from the initial states of the system. In other words, we wish to know whether there exists a sequence of transitions starting from the initial graphs which leads to the target graphs of the system.

In the next chapter, we take the specific example of a simple library system which consists of entities whose instances are related to each other. The instances and the relations between them can be modeled as a graph where the instances form the nodes of the graph and the relations between them form the edges. This is a typed graph in
which the type of a node is the entity of the instance which the node represents. There are no types for edges. Operations are performed in the system change the relations between instances of entities, i.e. operations perform graph transformations. We then introduce some safety properties for the library system which we would want the latter to satisfy.

1.2 Earlier Work On Graph Transformation Systems

Graph Grammars have been used to generate graph transformation systems. Graph grammars originated in the late 60s, motivated by considerations about pattern recognition and compiler construction. Since then the list of areas which have interacted with the development of graph grammars has grown considerably. Besides the aforementioned areas it includes software specification and development, database design, modeling of concurrent systems, and computer animation to name a few. The area of graph grammars and graph transformations generalizes formal language theory based on strings and the theory of term rewriting based on trees. Analogous to string grammars, a graph grammar $GG$ is defined as $GG = (G_0, P)$ where $G_0$ is a set of initial graphs and $P$ is a set of productions which specify how graphs are transformed. The language of the graph grammar is, just as for string grammars, defined as $L(GG) = \{G | G_0 \rightarrow^* G\}$ where $\rightarrow^*$ means zero or more applications of the productions.

Two approaches are used in graph grammars to define the productions. In the double pushout approach [CMR97], each production is defined as $L \leftarrow K \rightarrow R$. Here $L, K$ and $R$ are graphs and $K$ is a subgraph of $L$ and $R$. To apply this rule to a graph $G$, $L$ must be a subgraph of $G$. Then the part of $G$ contained in $L$ but outside $K$ (i.e. $L - K$) is deleted to produce a context graph $(G - L + K)$. Then the parts of $R$ outside $K$ $(R - K)$ are added to the context graph to produce the final graph which completes the application of the production to $G$. The 2 steps of the application of the production are called pushouts and hence the name double pushout. In the single pushout approach, each production is defined as $L \rightarrow R$. To apply this rule to a graph $G$, $L$ must be a subgraph of $G$. Then application of the rule involves deletion of the subgraph $L$ and replacing it with $R$ (i.e. the rule application produces $G - L + R$). Since there is just a single rewriting step, the approach is called the single pushout. This approach generalises the double-pushout approach [L93].

Tools have been developed based on the above approaches. AGG [LB93] is a general development environment for algebraic graph transformation systems which follows the interpretative approach. Its special power comes from a very flexible attribution concept. AGG graphs are allowed to be attributed by any kind of Java objects. Graph transformations can be equipped with arbitrary computations on these Java objects described by a Java expression. The AGG environment consists of a graphical user interface comprising several visual editors, an interpreter, and a set of validation tools. The interpreter allows the stepwise transformation of graphs as well as rule applications as long as possible. AGG supports several kinds of validations which comprise graph parsing, consistency checking.
of graphs and conflict detection in concurrent transformations by critical pair analysis of graph rules. Applications of AGG include graph and rule-based modeling of software, validation of system properties by assigning a graph transformation based semantics to some system model, graph transformation based evolution of software, and the definition of visual languages based on graph grammars.

PROGRES[97] is an integrated environment for a very high level programming language which has a formally defined semantics based on "PROgrammed Graph Rewriting Systems". This language supports the following programming paradigms/purposes among others: 1) Structurally object-oriented specification of attributed graph structures with multiple inheritance hierarchies and types of types (for parametric polymorphy). 2) Declarative/relational specification of derived attributes, node sets, binary relationships (directed edges), and Boolean constraints. Therefore, PROGRES can be used as 1) a very high level programming language for implementing abstract data types with a graph-like internal structure, 2) a rule-oriented language for rapid prototyping nondeterministically specified data/rule base transformations.
Chapter 2

The Library System

![UML Class Diagram for the Library System]

Figure 2.1: UML Class Diagram for the Library System

We first describe the library system.

The library system consists of members and books. A member can borrow a book kept on a shelf. In such a case the library loans the book to the member. A member can reserve a book by making a claim for title of the book. If a book for the claimed title exists on shelf, it is immediately reserved for the member, else the claim is kept pending until the book becomes available. A member can also borrow a book previously claimed by him. When the member returns a loaned book, the book goes on shelf if there are no pending claims for the title of the book. Otherwise, the book gets reserved. The library system also provides for adding more books to the library and more members to the system.
From the above description we can identify *entities* of the library system and the relations between them. These are expressed formally in a UML Class Diagram shown above.

The boxes represent the entities and the lines between them represent the *associations* between them. An association between entities A and B specifies a relation from A to B and one from B to A. It also specifies a *multiplicity value* for each of these relations. For example, **isIn** is a relation from Book to Shelf and the multiplicity 0..1 means that when an instance of Book exists, it can be related to at most one instance of Shelf (A book needn’t be on any shelf or it can be on only one shelf). But the **contains** relation having multiplicity 0..* allows an instance of Shelf to be related to any number of instances of books (A shelf can hold zero or more number of books). A multiplicity of 1 associated with the **hasTitle** relation means every instance of Book must be related to exactly one instance of a Title (A book must have exactly 1 title).

If we were to graphically depict the instances of the entities and the relations between them, we can see the library system as a graph having different kinds of nodes. Shown below is a library system having 5 books and 3 members.

![Figure 2.2: An example library system](image)

The 2 books with titles $t_1$ namely $b_3$ and $b_2$ are respectively loaned to and reserved by members $m_1$ and $m_2$. Member $m_3$ has borrowed book $b_4$ having title $t_2$ and has a pending claim for title $t_1$. The other 2 books $b_1$ and $b_5$ are on shelves $s_2$ and $s_1$ respectively. $c_1$ and $c_2$ are the claim nodes while $l_1$ and $l_2$ are the loan nodes.

We can also identify *operations* taking place in the system, from our description above. We call these operations **Borrow, Reserve, Return, Addbook** and **Addmember**. With these
operations taking place on the library system shown above, the graph can surely change. This motivates us to define the state of the library system in terms of the instances of entities and the relations between them.

We shall use First Order (FO) Logic henceforth in describing the library system and also later for formalising graph transformation systems. The problem of reachability will be formulated using FO. Note that though we have used a UML diagram to express the Library system, we are not trying to capture the semantics of the UML diagram using FO or provide any general way of capturing the semantics of UML diagrams using FO. The UML diagram just serves to show, as described in the next section, the different kinds of nodes and edges that exist in the state of the library system.

2.1 State of the library system

The entities of the library system are Book($\mathcal{B}$), Claim($\mathcal{C}$), Loan($\mathcal{L}$), Member($\mathcal{M}$), Shelf($\mathcal{S}$) and Title($\mathcal{T}$). Let $\mathcal{E} = \{\mathcal{B}, \mathcal{C}, \mathcal{L}, \mathcal{M}, \mathcal{S}, \mathcal{T}\}$. We define a state $s$ in terms of the following predicates:

- **Node predicates:**
  - $N(x)$ - This predicate is true iff the node $x$ is present in the graph of $s$.
  - $X(x), X \in \mathcal{E}$ - This predicate is true iff node $x$ represents an instance of entity $X$.

- **Edge predicates:**
  - $E(x,y)$ - This predicate is true iff an edge is present between nodes $x$ and $y$.

(There are no types for the edges).

Let $\sigma_s = \{N\} \cup \{E\} \cup \{X | X \in \mathcal{E}\}$.

Then the predicates of $\sigma_s$ completely define state $s$. Note that since there is no bound on the number of nodes in any state, the state space of the library system is infinite.

2.2 Operations of the Library System

Operations cause changes of state. Each operation takes as input some parameters and produces different graph transformations on the state under different conditions. These conditions are called guards and the transformations are called actions. An operation is expressed as a set of Guard-Action (GA) pairs. We now look at each of the operations, their semantics and their formal description in terms of GA pairs. An operation causes a transition from a state $s$ to a state $s'$. We describe state $s$ using the predicates of $\sigma_s$ subscripted with $s$ and describe state $s'$ using the predicates of $\sigma_s$ subscripted with $s'$. 
Thus for state $s$, we have $\sigma_s = \{N_s\} \cup \{E_s\} \cup \{X_s| X \in \mathcal{E}\}$ and for state $s'$, we have, $\sigma_{s'} = \{N_{s'}\} \cup \{E_{s'}\} \cup \{X_{s'}| X \in \mathcal{E}\}$.

1. $\text{Addbook}(b, t, u)$:
This operation adds a Book instance $b$ having title as Title instance $t$ to shelf denoted by Shelf instance $u$ if there are no pending claims for the book. If there are pending claims, the book gets reserved.

Viewed as a graph transforming operation, this operation adds a node for $b$ and nodes for $t$ and $u$ if they are already not present in the current state. If there is no Claim node connected to $t$, the $b - t$ and $t - u$ edges are added, else the $c - t$ edge is deleted and the $c - b$ and $b - t$ edges are added. We can express the GA pairs for the operation as shown below.

Guard 1 below expresses that $b$ is a Book node and is absent in state $s$ (this is expressed by $B_s(b) \land \neg N_s(b)$), $t$ is a Title node and is present in state $s$ ($T_s(t) \land N_s(t)$), $u$ is a Shelf node and is present in state $s$ ($S_s(u) \land N_s(u)$). $\neg \exists c(C_s(c) \land N_s(c) \land E_s(c, t))$ expresses that there is no Claim node present in state $s$ which has an edge to the node $t$. The changes produced by the action involve addition of $b$ to $s'$ without changing its type ($B_{s'}(b) \land N_{s'}(b)$) and the addition of the $b - t$ and $b - u$ edges ($E_{s'}(b, t) \land E_{s'}(b, u)$).

The other GA pairs can be understood similarly.

### Guard

<table>
<thead>
<tr>
<th></th>
<th>Changes produced by Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$B_s(b) \land T_s(t) \land S_s(u)$</td>
</tr>
<tr>
<td></td>
<td>$\neg N_s(b) \land N_s(t) \land N_s(u)$</td>
</tr>
<tr>
<td></td>
<td>$\land \neg \exists c(C_s(c) \land N_s(c) \land E_s(c, t))$</td>
</tr>
<tr>
<td>2</td>
<td>$B_s(b) \land T_s(t) \land S_s(u)$</td>
</tr>
<tr>
<td></td>
<td>$\neg N_s(b) \land \neg N_s(t) \land N_s(u)$</td>
</tr>
<tr>
<td></td>
<td>$\land \neg \exists c(C_s(c) \land N_s(c) \land E_s(c, t))$</td>
</tr>
<tr>
<td>3</td>
<td>$B_s(b) \land T_s(t) \land S_s(u)$</td>
</tr>
<tr>
<td></td>
<td>$\neg N_s(b) \land N_s(t) \land \neg N_s(u)$</td>
</tr>
<tr>
<td></td>
<td>$\land \neg \exists c(C_s(c) \land N_s(c) \land E_s(c, t))$</td>
</tr>
<tr>
<td>4</td>
<td>$B_s(b) \land T_s(t) \land S_s(u)$</td>
</tr>
<tr>
<td></td>
<td>$\neg N_s(b) \land \neg N_s(t) \land \neg N_s(u)$</td>
</tr>
<tr>
<td></td>
<td>$\land \neg \exists c(C_s(c) \land N_s(c) \land E_s(c, t))$</td>
</tr>
<tr>
<td>5</td>
<td>$B_s(b) \land T_s(t)$</td>
</tr>
<tr>
<td></td>
<td>$\neg N_s(b) \land N_s(t)$</td>
</tr>
<tr>
<td></td>
<td>$\land \exists c(C_s(c) \land N_s(c) \land E_s(c, t))$</td>
</tr>
</tbody>
</table>
In the last GA pair, if there are many \( c - t \) edges, the operation deletes exactly one \( c - t \) edge and leaves others unchanged. Hence in case of many \( c - t \) edges, there are as many next states (produced by choosing different \( c - t \) edges to delete).

The action above only describes the modifications to the state \( s \). Next state \( s' \) is described completely by considering the unaffected edges and nodes of \( s \) too.

2. **Addmember(\( m \)):**
   This operation adds \( m \) provided it is not already present.

   \[
   \text{Guard} \quad \text{Changes produced by Action} \\
   1. \quad \mathcal{M}_s(m) \land \neg N_s(m) \quad \mathcal{M}_s(m) \land N_s(m)
   \]

3. **Borrow(\( b,m \)):**
   Book \( b \) is loaned to member \( m \) only if either the book is on shelf or it was previously reserved by \( m \).

   Guard 1 below expresses that \( b \) is a Book node present in state \( s \) \((\mathcal{B}_s(b) \land N_s(b))\) and \( m \) is a Member node present in state \( s \) \((\mathcal{M}_s(m) \land N_s(m))\). \( \exists l(\mathcal{L}_s(l) \land \neg N_s(l)) \land \exists u(\mathcal{S}_s(u) \land N_s(u) \land E_s(b, u)) \) expresses that there is a Loan node \( l \) which is absent in \( s \) and there is a Shelf node \( u \) present in \( s \) which has an edge to the \( b \) node. Then the changes produced by the action involve addition of \( l \) to \( s' \) without changing its type \((\mathcal{L}_{s'}(l) \land N_{s'}(l))\), the addition of the \( b - l \) and \( l - m \) edges \((E_{s'}(b, l) \land E_{s'}(l, m))\) and the deletion of the \( b - u \) edge \((\neg E_s(b, u))\). GA pair 2 can be understood similarly.

   \[
   \text{Guard} \quad \text{Changes produced by Action} \\
   1. \quad \mathcal{B}_s(b) \land \mathcal{M}_s(m) \land N_s(b) \land N_s(m) \land \exists l(\mathcal{L}_s(l) \land \neg N_s(l)) \land \exists u(\mathcal{S}_s(u) \land N_s(u) \land E_s(b, u)) \quad \mathcal{L}_{s'}(l) \land N_{s'}(l) \land E_{s'}(b, l) \land E_{s'}(l, m) \land \neg E_{s'}(b, u) \\
   2. \quad \mathcal{B}_s(b) \land \mathcal{M}_s(m) \land N_s(b) \land N_s(m) \land \exists l(\mathcal{L}_s(l) \land \neg N_s(l)) \land \exists c(\mathcal{C}_s(c) \land N_s(c) \land E_s(c, m) \land E_s(c, b)) \quad \mathcal{L}_{s'}(l) \land N_{s'}(l) \land E_{s'}(b, l) \land E_{s'}(c, b) \land \neg E_{s'}(c, m)
   \]

4. **Reserve(\( t,m \)):**
   If a book for the title \( t \) is present on shelf, it is reserved immediately. Else the claim is kept pending.

   Guard 1 below expresses that \( t \) is a Title node present in state \( s \) \((\mathcal{T}_s(t) \land N_s(t))\) and \( m \) is a Member node present in state \( s \) \((\mathcal{M}_s(m) \land N_s(m))\). \( \exists c(\mathcal{C}_s(c) \land \neg N_s(c)) \land \neg \exists b \exists u(\mathcal{B}_s(b) \land \mathcal{S}_s(u) \land N_s(b) \land N_s(u) \land E_s(b, t) \land E_s(b, u)) \) expresses that there is a Claim node \( c \) which is absent in \( s \) and there is no Book node \( b \) and Shelf node \( u \) which are present in \( s \) and such that the \( b - u \) and \( b - t \) edges are present. The changes produced by the action involve addition of \( c \) to \( s' \) without changing its type \((\mathcal{C}_{s'}(c) \land N_{s'}(c))\), the addition of the \( c - m \) and \( c - t \) edges \((E_{s'}(c, m) \land E_{s'}(c, t))\). GA pair 2 can be understood similarly.
5. Return(b,m):
If book b is loaned to member m, then if there are no pending claims, the book is returned to shelf if present or a new shelf is created for the returned book. If there are pending claims for the title of the book, the book gets reserved.

Guard 1 below expresses that b is a Book node present in state s (B_s(b) ∧ N_s(b)) and m is a Member node present in state s (M_s(m) ∧ N_s(m)). ∃u(M_s(m) ∧ N_s(m)) ∧ ∃(L(l) ∧ N_s(l) ∧ E_s(b, l) ∧ E_s(l, m)) expresses that there is a Shelf node u which is present in s and there is a Loan node l present in s which has edges to the b and m nodes. ¬∃t∃c(T_t(t) ∧ C_s(c) ∧ N_s(t) ∧ N_s(c) ∧ E_s(b, t) ∧ E_s(c, t)) expresses there is no Title node t and Claim node c which are present in s and such that the b − t and c − t edges are present. The changes produced by the action involve deletion of l to s’ without changing its type (L_s’(l) ∧ ¬N_s’(l)) and deletion of all its edges (¬E_s’(b, l) ∧ ¬E_s’(l, m)) and the addition of the b − u edge (E_s’(b, u)). GA pairs 2 and 3 can be understood similarly.
The figure below shows how the transitions of the library system from one state to another are caused by operations taking place in the library system. Each edge is labelled with the operation and the specific GA pair of the operation involved in the transition. The library system starts from the empty graph.

### 2.3 Initial State Conditions and Safety Properties

The library system starts from the empty state where there are no instances of any entity. Using the predicates of $\sigma_S$, we can write the initial condition as

$$\forall x \neg N_s(x) \land \forall x, y \neg E_s(x, y)$$

The system evolves through the operations. While we note that a state of the system is indeed a graph, we also note that the nodes in the graph cannot be arbitrarily connected. For example, the UML diagram disallows any edge between a Shelf instance and a Loan.
instance or between a Member instance and a Title instance. Similarly, no Book instance must be connected to both a Loan instance and a Shelf instance otherwise it would mean that the book is simultaneously on shelf and loaned to a member. There are thus several patterns of interconnections between entities that the operations must not produce. We shall consider the specific problem of checking for the $\mathcal{C}$-$\mathcal{T}$-$\mathcal{B}$-$\mathcal{S}$ pattern. This pattern is present in a state if there is a Claim instance connected to a Title instance which is connected to a Book instance which in turn is connected to a Shelf instance. If this pattern is present it means that a claim for a title is kept pending even when there is a book for the title on shelf - clearly an undesirable situation. Using the predicates of $\sigma_S$, we can express this as a safety property as

$$
\neg \exists c, t, b, s \quad (C_s(c) \land T_s(t) \land B_s(b) \land S_s(s) \land N_s(c) \land N_s(t) \land N_s(b) \land N_s(s) \land
E_s(c, t) \land E_s(b, t) \land E_s(b, s))
$$

We shall see later how we can check this safety property (and several other similar properties) in the library system.

While it is known that reachability analysis in infinite state systems is undecidable in general, in the next chapter, we show that reachability analysis in infinite state graph transformation systems is also, in general, undecidable.
Chapter 3

Undecidability of Reachability in Infinite State Graph Transforming Systems

Undecidability of reachability is well-known for graph-transforming systems produced by graph rewrite systems. Though we have to formally investigate how GTSes, the way we have described them, compare with GTSes produced by graph rewrite systems, there are parallels between the two and hence undecidability of reachability in our GTSes can be expected.

We show a reduction from the halting problem. The halting problem asks whether given a Turing machine $M$ and a string $w$, $M$ halts on $w$. The problem is known to be undecidable.

Let $M = (Q, \Sigma, \Gamma, q_1, H, \delta)$ be the given Turing machine $M$ where $Q$ is the set of states, $\Sigma = \{a_1, a_2, \ldots, a_r\}$ is the string alphabet, $\Gamma = \Sigma \cup \{b\}$ is the tape alphabet ($b$ is the blank symbol), $q_1$ is the initial state, $H \subseteq Q$ is the set of halting states and $\delta : Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L, R\}}$ is the transition function ($L, R$ denote left and right movement of the head respectively).

We consider the tape of $M$ to be semi-infinite.

We construct a graph-transformation system $S$ whose state space is the space of all possible configurations of $M$ represented as graphs. Each configuration of $M$ can be described as $y_1 y_2 \ldots y_{m-1} q_i y_m y_{m+1} \ldots y_n$ where $y_1 y_2 \ldots y_n$ denotes the string on the tape with $y_n$ being the rightmost non-blank symbol, $q_i$ denotes the state $M$ is in and $y_m$ denotes the symbol under the head.

Then define the set of types for nodes to be

$$N_t = \{A_1, A_2, \ldots, A_r, B, C, Q_1, Q_2, \ldots, Q_k\}$$

where the $A_i$’s correspond to $\Sigma$, the $Q_i$’s correspond to the states of $Q$, $B$ corresponds to the blank symbol and $C$ can be regarded as corresponding to the control.

Then the graph corresponding to the above configuration is shown below:
Figure 3.1: Graph corresponding to the configuration \( y_1 \ldots y_m q_i y_m \ldots y_n \)

Here \( q_i (1 \leq i \leq k) \) is a node of type \( Q_i \) and \( c \) is of type \( C \). \( x_i (1 \leq i \leq n) \) is of type \( A_j \) if \( y_i = a_j \). \( x_i \) is of type \( B \) if \( y_i = b \).

For each transition \( \{q'_i, a'_i, \rho\} \in \delta(q_i, a_i) \) (\( \rho \in \{L, R\} \)), we construct an operation which has no parameters and which has just one Guard-Action (GA) pair as shown below:

- **Guard**
  - Check if there is \( x_m \) (of type \( A_i \)) such that the edge \( c \rightarrow x_m \) is present. Check if there is \( q_i \) (of type \( Q_i \)) such that the edge \( c \rightarrow q_i \) is present.

- **Action**
  1. Delete the edge \( c \rightarrow q_i \) and add the edge \( c \rightarrow q'_i \).
  2. Delete \( x_m \) and all its incoming and outgoing edges and add an instance \( x'_m \) of \( A'_i \).
  3. If \( x_m \) had an outgoing edge to \( x_{m+1} \), add the edge \( x'_m \rightarrow x_{m+1} \). If \( x_m \) had an incoming edge from \( x_{m-1} \), then add the edge \( x_{m-1} \rightarrow x'_m \). (This step corresponds to the overwriting of the symbol \( a_i \) with \( a'_i \).)
  4. \( \rho = L \):
     - If \( x_m \) had an incoming edge from \( x_{m-1} \), add the edge \( c \rightarrow x_{m-1} \). Else add the edge \( c \rightarrow x'_m \).
     (This step corresponds to the movement of the head - if the symbol \( y_m \) corresponding \( x_m \) was not the leftmost symbol, the tape head is moved left so that it is now over \( y_{m-1} \). Else the head stays where it is.)
   - \( \rho = R \):
     - If \( x_m \) had an outgoing edge to \( x_{m+1} \), add the edge \( c \rightarrow x_{m+1} \). Else add an instance \( x_{m+1} \) of \( B \) and add the edges \( x'_m \rightarrow x_{m+1} \) and \( c \rightarrow x_{m+1} \).
     (This step corresponds to the movement of the head - if the symbol \( y_m \) corresponding \( x_m \) was not the rightmost non-blank symbol, the tape head is moved right so that it is now over \( y_{m+1} \). Else the head moves right and is over a blank symbol.)
The graph transformation system is then described as the set of all operations as constructed above for all the transitions in $\bigcup_{q \in Q, a \in \Sigma} \delta(q, a)$.

In the graph transformation system, the set of all next graphs for a given current graph is obtained by considering together the set of next graphs produced by each operation for each possible value of its parameters. Since each operation constructed as above has no parameters and has only one GA pair, for the (current) graph corresponding to the current configuration $y_1 \ldots y_{m-1}qa \ldots y_n$, the operation for the transition $(q', a', \rho) \in \delta(q, a)$ produces a unique next graph and this graph corresponds to the configuration obtained by taking the transition $(q', a', \rho)$ from the current configuration. Then we can see that, for the (current) graph (corresponding to the current configuration of $M$), the set of all next graphs corresponds precisely to the set of all next configurations of $M$ obtained by one move of $M$ from the current configuration.

The initial graph is the graph corresponding to the initial configuration. If the initial configuration of $M$ is $q_1y_1y_2 \ldots y_n$, then the initial graph is as shown below.

![Graph diagram](image_url)

Figure 3.2: Graph corresponding to the initial configuration $q_1y_1 \ldots y_n$

We now wish to check the reachability of the halting graphs in which the $c \rightarrow q_h$ edge is present where $c$ is of type $C$ and $q_h$ is of type $Q_h$, where $Q_h$ is any type corresponding to the halting states $H$. Clearly $M$ halts on $w$ iff any halting graph is reachable from the initial graph and the reduction is clearly a polynomial one (In fact it is a linear reduction).
Chapter 4

Reachability Analysis Using Induction

We now look at how reachability analysis can be done using induction in a general infinite state transition system.

Consider an infinite state system whose set of states we denote by $S$. Let $I$ be a predicate on $S$ representing all initial states. $T$ is the transition relation which is a predicate on $S \times S$. We wish to check if a safety property $P$, which is a predicate on $S$, is true in all the reachable states of the system. Presented below is a procedure towards this end. This procedure draws inspiration from [MSG00] which addresses the same problem, but for a finite state system.

We first introduce some notations before explaining procedure. The notations are the same as in [MSG00]. We denote a sequence of states $s_1, s_2, \ldots, s_k$ by $s_{[1..k]}$. We use $\text{path}(s_{[1..k]})$ to express that $s_{[1..k]}$ forms a path in the system. Thus

$$\text{path}(s_{[1..k]}) = \bigwedge_{i=1}^{i=k} T(s_i, s_{i+1})$$

While $\text{path}(s_{[1..k]})$ expresses that $s_{[1..k]}$ forms a path in the system, it does not tell us whether the path has loops or not. Loop-free paths play an important part in our procedure below, so we introduce $\text{loopfree}(s_{[1..k]})$ to express that $s_{[1..k]}$ is a loopfree path in the system. Thus

$$\text{loopfree}(s_{[1..k]}) = \text{path}(s_{[1..k]}) \land \bigwedge_{1 \leq i < j \leq k} s_i \neq s_j$$

We note that for any $i, 1 \leq i \leq k$,

$$\text{loopfree}(s_{[1..k]}) \rightarrow \text{loopfree}(s_{[1..i]}) \land \text{loopfree}(s_{[i+1..k]})$$

Finally to express that a property $Z$ is true in all the states of the sequence $s_{[1..k]}$, we introduce $\text{all}.Z(s_{[1..k]})$ defined as
We now look at the procedure InductionSolve below. Valid is a procedure which takes as input a formula $F(s_1, s_2, \ldots, s_n)$ which is a boolean combination of the formulae $I(s_i)(1 \leq i \leq n), P(s_i)(1 \leq i \leq n), T(s_i, s_j)(1 \leq i, j \leq n)$ and $s_i = s_j(1 \leq i, j \leq n)$ returns True if and only if the $F(s_1, s_2, \ldots, s_n)$ is true for all $(s_1, s_2, \ldots, s_n) \in S^n$ i.e. if $F$ is valid.

4.1 InductionSolve

1. If (Valid ($P(s)$))
   return True
2. else if (not(Valid ($I(s) \rightarrow P(s)$)))
   return counterexample $s$.
3. else
4. \hspace{2em} k := 1
5. \hspace{2em} while True do
6. \hspace{4em} If (Valid ($loopfree(s_{[1\ldots k+1]} \land all.P(s_{[1\ldots k]}) \rightarrow P(s_{k+1})$))
   \hspace{6em} return True
7. \hspace{4em} else if (not(Valid ($I(s_{1}) \land loopfree(s_{[1\ldots k+1]} \land all.\neg I(s_{[2\ldots k+1]} \land all.P(s_{[1\ldots k]}) \rightarrow P(s_{k+1})$)))
   \hspace{6em} return counterexample $s_{[1\ldots k+1]}$
8. \hspace{4em} else
9. \hspace{6em} k := k+1
10. done

\hspace{2em} InductionSolve

4.2 Intuitive Idea behind InductionSolve

The intuitive idea of the procedure is as follows:

We first check whether $P$ is true in all states of the system(line 1). If so, it is certainly true in all the reachable states and we are done. This is the base case of induction process. If this fails, then $P$ is violated in some state of the system and we wish to know if it is reachable. The most basic step is to check whether the state is already an initial state(line 2). If so, we have a counterexample. Else all the initial states are $P$-states (i.e. satisfy $P$). We then take the inductive step of checking whether from any $P$-state we can go only to another $P$-state(line 6, $k = 1$) in one step. If this is true, then all the reachable states would be $P$-states and we would be done. Else there is some $P$-state leading to a $\neg P$-state in one step and we wish to know whether it is reachable. Again we check whether this state is an initial state in the first place(line 7, $k = 1$). If so, we have a path to a state
violating $\mathcal{P}$. Else all the neighbours of the initial states are also $\mathcal{P}$-states. We then take the next inductive step of checking that in a sequence of two steps is it true that if the first two states are $\mathcal{P}$-states then the third is also a $\mathcal{P}$-state (line 6, $k = 2$). The subsequent steps are then similar to the first inductive step explained above.

### 4.3 Proof Of Correctness

We now formally prove the correctness of the procedure (whenever it terminates). If the procedure returns a counterexample then it means that for some $k$, these are states $s_1, s_2 \ldots s_{k+1}$ such that

$$\mathcal{I}(s_1) \land \text{loopfree}(s_{[1..k+1]}) \land \neg \mathcal{P}(s_{k+1})$$

is true. This sequence of states actually gives the path starting from the initial state to a reachable state $(s_{k+1})$ violating $\mathcal{P}$.

We look at the case when the procedure returns True. We prove that

$$\mathcal{I}(s_1) \land \text{loopfree}(s_{[1..i]}) \land \neg \mathcal{P}(s_i)$$

is contradictory $\forall i \in \{1, 2, \ldots\}$ thus showing that if a state $s$ satisfies $\neg \mathcal{P}(s)$, then it is not reachable from the initial states, so that all the reachable states of the system satisfy the safety property $\mathcal{P}$.

**Proof:**

Firstly we note that the procedure returns True either from line 1 or line 6. If procedure returns True from line 1 it means $\mathcal{P}(s)$ is true. Thus $\mathcal{I}(s) \land \neg \mathcal{P}(s)$ is contradictory. Let us now look at the more interesting case when procedure returns True from line 6. Then in that case, for some $k \geq 1$

1. $A(s_{[1..k+1]}) = \text{loopfree}(s_{[1..k+1]}) \land \text{all.}\mathcal{P}(s_{[1..k]}) \land \neg \mathcal{P}(s_{k+1})$
   is contradictory and

2. $\forall i \in \{1, 2, \ldots k - 1\}$
   $B(s_{[1..k+1]}) = \mathcal{I}(s_1) \land \text{loopfree}(s_{[1..k+1]}) \land \text{all.}\neg \mathcal{I}(s_{[2..k+1]}) \land$
   $\text{all.}\mathcal{P}(s_{[1..k]}) \land \neg \mathcal{P}(s_{k+1})$
   is contradictory.

We make the important observation that, that the procedure returns True from line 6 itself means that $\text{Valid}(\mathcal{I}(s) \rightarrow \mathcal{P}(s))$ returned True in line 2 i.e. all initial states are $\mathcal{P}$-states (i.e. they satisfy $\mathcal{P}$).

We consider two cases
1. $1 \leq i \leq k$

Consider the following subcases:

(a) $I(s_1) \land \text{loopfree}(s_{[1..i]}) \land \neg \forall i. I(s_{[2..i]}) \land \forall i \neg P(s_{[1..i-1]}) \land \neg P(s_i)$

If this is satisfiable, it means there is a sequence of states $s_{[1..i]}$ starting from initial state $s_1$ ending in $\neg P$ states $s_i$, such that some intermediate states are also initial states. Then consider the last initial state $s_j$ ($1 < j < i$). (Note that the $\neg P$ state cannot be an initial state). Then the subsequence $s_{[j..i]}$ is a sequence $s_{[1..i']}$ of length $i' = i - j + 1 < k$ such that $B(s_{[1..i']})$ is true. As seen above, this is not possible.

(b) $I(s_1) \land \text{loopfree}(s_{[1..i]}) \land \forall i. \neg I(s_{[2..i]}) \land \forall i \neg P(s_{[1..i-1]}) \land \neg P(s_i)$

If this is satisfiable, then consider the first state $s_j(1 < j < i \leq k)$ violating $P$. Then $B(s_{[1..j]})$ is false which is not possible.

(c) $I(s_1) \land \text{loopfree}(s_{[1..i]}) \land \neg \forall i. I(s_{[2..i]}) \land \forall i \neg P(s_{[1..i-1]}) \land \neg P(s_i)$

If this is satisfiable, then consider the first state $s_j(1 < j < i \leq k)$ violating $P$. If there is no intermediate initial state in the subsequence $s_{[1..j]}$, then $B(s_{[1..j]})$ is true which is not possible. If there is, then by case (a), we again have a contradiction.

Putting together the subcases with (2) above, we can see that $I(s_1) \land \text{loopfree}(s_{[1..i]}) \land \neg P(s_i)$ is indeed contradictory.

2. $i > k$

We consider the following subcases.

(a) $I(s_1) \land \text{loopfree}(s_{[1..i]}) \land \forall i. \neg I(s_{[2..i]}) \land \forall i \neg P(s_{[1..i-1]}) \land \neg P(s_i)$

If this is satisfiable, then the subsequence $s_{[i-k..i]}$ is a subsequence $s_{[1..i+k+1]}$ such that $A(s_{[1..i+k+1]})$ is true, which is not possible. Thus $B(s_{[1..i]})$ is false for all $i$.

(b) $I(s_1) \land \text{loopfree}(s_{[1..i]}) \land \neg \forall i. I(s_{[2..i]}) \land \forall i \neg P(s_{[1..i-1]}) \land \neg P(s_i)$

Then let the last initial state be $s_j(1 < j < i)$. Then the subsequence $s_{[j..i]}$ is a sequence $s_{[1..i']}$ of some length $i' \geq 1$ for which $B(s_{[1..i']})$ is true which we have already shown to be impossible in (a). Thus the above formula is false for all $i$.

(c) $I(s_1) \land \text{loopfree}(s_{[1..i]}) \land \forall i. \neg I(s_{[2..i]}) \land \forall i \neg P(s_{[1..i-1]}) \land \neg P(s_i)$

Then consider the first state $s_j(1 < j < i)$ which violates $P$. Then $B(s_{[1..j]})$ is true. Contradiction. This formula too is thus false for all $i$.

(d) $I(s_1) \land \text{loopfree}(s_{[1..i]}) \land \neg \forall i. I(s_{[2..i]}) \land \forall i \neg P(s_{[1..i-1]}) \land \neg P(s_i)$

Again consider the first state $s_j(1 < j < i)$ which violates $P$. If there is no initial state in $s_{[2..j]}$, the formula is false by subcase (a), else it is false by subcase (b). Thus the formula is false for all $i$.

Putting 2(a), (b), (c) and (d) together we see that the proof is complete.
4.3. PROOF OF CORRECTNESS

Thus in InductionSolve, we use a combination of inductive reasoning (line 6) and bounded model checking (line 7) to either prove that the all reachable states are $\mathcal{P}$ states or to find a path from the initial states to a property violating state.

We now note that since reachability in infinite state systems is undecidable in general, the above procedure is not guaranteed to terminate always. There are two ways of non-termination of InductionSolve:

1. An unbounded increase of the iterator $k$
2. Non-termination of the validity checker

We show now how even if the validity checker is a terminating one, we cannot avoid non-termination due the an unbounded increase of iterator $k$.

Consider the simple graph transformation system described in the previous section. We saw that a graph(state) of the graph transformation system corresponds to a configuration of the TM. The initial graph corresponds to the initial configuration and the property violating graphs correspond to the halting configurations of the TM. Further, for a graph $s$, the set of next graphs corresponds exactly to the set of next configurations of the TM for the current configuration corresponding to $s$. Then the validity checker does the following:

1. If the input is $\mathcal{P}(s)$ (line 1), the validity checker has to check whether at all there is a graph violating $\mathcal{P}(s)$ in the graph transformation system, i.e. whether at all there is a halting configuration. Clearly the space of all possible configurations includes the halting configuration too and hence $\neg \mathcal{P}(s)$ is satisfiable (where the satisfying assignment is the graph corresponding to the halting configuration) and hence the validity checker returns False.

2. If the input is $\mathcal{I}(s) \rightarrow \mathcal{P}(s)$ (line 2), the validity checker has to check whether the initial configuration is a halting configuration. This is done by simply checking if $q_1 \in H$. If yes, the validity checker returns False else it returns True.

3. If the input is

\[
A = (\text{loopfree}(s_{[1..k+1]}) \land \text{all}.\mathcal{P}(s_{[1..k]})) \rightarrow \mathcal{P}(s_{k+1}) \text{ or }
B = (\mathcal{I}(s_1) \land \text{loopfree}(s_{[1..k+1]}) \land \text{all}.\neg\mathcal{I}(s_{[2..k+1]}) \land \text{all}.\mathcal{P}(s_{[1..k]})) \rightarrow \mathcal{P}(s_{k+1}),
\]

(lines 6,7) the validity checker has to determine if there exists a sequence of $k + 1$ configurations $s_1, s_2, \ldots, s_{k+1}$ of the TM such that the TM moves from $s_i$ to $s_{i+1}(1 \leq i \leq k)$ and such that only $s_{k+1}$ is a halting configuration (and in case of $B$ whether $s_1$ is the only initial state). Now in $k$ moves, the TM can move only atmost $k$ cells to the right or $k$ cells to the left on the tape. The validity checker then considers a tape of length $2k + 1$. Then the number of configurations $s_1$ is $u = |Q| \times (2k + 1) \times |\Gamma|^{2k+1}$. Now for each configuration, the number of next configurations of the TM is $\max\{|\delta(q,a)| \mid q \in Q, a \in \Gamma\}$ and this is atmost $v = |Q| \times |\Gamma| \times 2$. Then the
number of sequences of configurations $s_1, s_2, \ldots, s_{k+1}$ is atmost $u \times v^k$. The validity checker enumerates out all the possible sequences of configurations $s_1, s_2, \ldots, s_{k+1}$ and checks if $A$ resp. $B$ is true in these sequences. If $A$ resp. $B$ is false for any sequence $s_1, s_2, \ldots, s_{k+1}$, then $s_1, s_2, \ldots, s_{k+1}$ is a sequence of configurations violating $A$ resp. $B$ and hence the validity checker returns False. If however $A$ resp. $B$ is true for all the above sequences $s_1, s_2, \ldots, s_{k+1}$, then there cannot exists a sequence of configurations $s_1, s_2, \ldots, s_{k+1}$ violating $A$ resp. $B$ because if there was such a sequence, then by considering only those cells of TM which were visited by the TM in the configurations $s_1, s_2, \ldots, s_{k+1}$ (the number of these cells being atmost $k+1$), we can construct a sequence of configurations $s'_1, s'_2, \ldots, s'_{k+1}$ belonging to the set of the $u\cdot v^k$ sequences considered above. But all of the latter sequences have already been checked by the validity checker and found to satisfy $A$ resp. $B$. Thus $s_1, s_2, \ldots, s_{k+1}$ cannot exist.

Thus for the graph transformation system of the previous section, we have a terminating validity checker. But we cannot put any bounds on the iterator $k$ and expect InductionSolve to terminate and give the correct answer because then we would be able to decide the halting problem. The unbounded increase of $k$ and hence the non-termination of InductionSolve can thus not be avoided in general.

The method of inductive reasoning and bounded model checking used in InductionSolve is, as we later found, very much similar to the method used in SAL(Symbolic Analysis Laboratory) SAL[NS00] is a framework for combining different tools for abstraction, program analysis, theorem proving, and model checking toward the calculation of properties (symbolic analysis) of transition systems. A key part of the SAL framework is an intermediate language for describing transition systems. This language serves as the target for translators that extract the transition system description for popular programming languages such as Esterel, Java, and Statecharts. The intermediate language also serves as a common source for driving different analysis tools through translators from the intermediate language to the input format for the tools, and from the output of these tools back to the SAL intermediate language. The main difference between InductionSolve and SAL is while the former uses a validity checker to do inductive reasoning or bounded model checking, the latter uses a theorem prover. Further, the latter may use human intervention during theorem proving. However by considering specific graph-transforming systems and safety properties, we can ensure that no user intervention is required for the validity checker.

In the next chapter, we look at a formalisation of graph transformation systems. We shall later use InductionSolve in conjunction with this formalisation to do reachability analysis in graph transforming systems.
Chapter 5

Logical Formalisation of Graph Transformation Systems

As seen earlier, Graph Transformation Systems are state-transition systems in which each state contains a graph and the transition from one state to another involves a graph transformation. The graphs we consider are typed graphs, i.e., each node and each edge has a type associated with it. The type of each node is a subset of a finite set of attributes \( N_t \) for nodes \( (|N_t| = N_n) \) and the type of each edge is a subset of a finite set of attributes \( E_t \) for edges \( (|E_t| = N_e) \). For ease of exposition, we henceforth regard the attributes themselves as types so that a node has a set of types associated with it and similarly an edge has a set of types associated with it.

Before proceeding to formally define a state, we shall first look at some preliminaries from mathematical logic. The notation is as used in [LL98].

5.1 Preliminaries from Mathematical Logic

A vocabulary \( \sigma \) is a collection of constant symbols \( c_1, \ldots, c_p \), predicate symbols \( P_1, \ldots, P_q \) and function symbols \( f_1, \ldots, f_r \). Each predicate and function symbol has an arity associated with it, which is the number of arguments/parameters of the symbol.

A \( \sigma \)-structure

\[
M = \langle A, \{c_1^M, \ldots, c_p^M\}, \{P_1^M, \ldots, P_q^M\}, \{f_1^M, \ldots, f_r^M\} \rangle
\]

has a universe \( A \) and interpretations of

1. each constant symbol \( c_i (1 \leq i \leq p) \) as an element \( c_i^M \in A \).
2. each predicate symbol \( P_i (1 \leq i \leq q) \) having arity \( k_i \) as a \( k_i \)-ary relation on \( A \), i.e., \( P_i^M \subseteq A^{k_i} \).
3. each function symbol $f_i(1 \leq i \leq r)$ having arity $k_i$ as a $k_i$-ary function from $A^{k_i}$ to $A$, i.e. $f_i^M : A^{k_i} \rightarrow A$.

A vocabulary is called relational if it contains only predicate symbols and constant symbols and no function symbols and is called purely relational if it contains only predicate symbols and no constant or function symbols.

We now look at First Order (FO) formulae, free and bound variables and semantics of FO formulae.

Variables are denoted as $x, y, z, \ldots$ with subscripts and superscripts. Then terms and FO formulae are defined as below.

- Each variable $x$ is a term.
- Each constant $c_i$ is a term.
- For a $k_i$-ary function symbol $f_i$, if $x_1, \ldots, x_{k_i}$ are variables, then $f_i(x_1, \ldots, x_{k_i})$ is a term.
- If $t_1, t_2$ are terms, then $t_1 = t_2$ is a formula (in particular, atomic formula).
- For predicate symbol $P_i$ of arity $k_i$ and variables $x_1, \ldots, x_{k_i}$, $P_i(x_1, \ldots, x_{k_i})$ is a formula (in particular, atomic formula).
- If $\varphi_1, \varphi_2$ are formulae, then any boolean combination of these is a formula.
- If $\varphi$ is a formula, $\exists x \varphi$ and $\forall x \varphi$ are formulae.

Free variables of a formula are as defined below:

- The free variable of term $x$ is $x$. The constant $c_i$ has no free variables.
- Free variables of $t_1 = t_2$ are the free variables of $t_1$ and $t_2$. Free variables of $P_i(x_1, \ldots, x_{k_i})$ and $f_i(x_1, \ldots, x_{k_i})$ are $x_1, \ldots, x_{k_i}$.
- Free variables of $\varphi_1 \land \varphi_2$ (and of $\varphi_1 \lor \varphi_2$) are the free variables of $\varphi_1$ and $\varphi_2$. Free variables of $\neg \varphi_1$ are the same as of $\varphi_1$.
- Free variables of $\exists x \varphi$ and $\forall x \varphi$ are the free variables of $\varphi$ excluding $x$.

The variables which are not free are bound. A formula $\varphi$ with free variables $x$ is denoted as $\varphi(x)$.

Given a $\sigma$-structure $M$, for each term $t$ with free variables $x_1, \ldots, x_n$, given below is the inductive definition of the value of $t^M(a)$ where $a \in A^n$. Similarly for a formula $\varphi(x_1, \ldots, x_n)$, we see below how the notion of $M \models \varphi(a_1, \ldots, a_n)$ ($\varphi(a_1, \ldots, a_n)$ is true in $M$) is defined.
5.2 Defining a State

A state \( s \) is a \( \sigma_s \)-structure where
\[
\sigma_s = \{ N \} \cup \{ E \} \cup \{ T_N^i | 1 \leq i \leq N_n \} \cup \{ T_E^i | 1 \leq i \leq N_e \}
\]
The elements of the universe of \( s \) are nodes.

1. \( N(x) \) denotes that the node \( x \) is present in \( s \).

2. \( E(x, y) \) denotes that the edge \((x, y)\) is present in \( s \).

3. As stated above, there is a finite set, \( N_t \) of types for nodes. \( |N_t| = N_n \). Each node has a set of types associated with it which is a subset of \( N_t \). Then \( T_N^i(x)(1 \leq i \leq N_n) \) denotes that the type \( i \) is present in the set of types of node \( x \).

4. As stated above, there is a finite set, \( E_t \) of types for edges. \( |E_t| = N_e \). Each edge has a set of types associated with it which is a subset of \( E_t \). Then \( T_E^i(x, y)(1 \leq i \leq N_e) \) denotes that the type \( i \) is present in the set of types of edge \((x, y)\).

The graph of \( s \) is then defined by the nodes and edges present in the \( s \) and by their types.
5.3 Defining A Graph Transformation System $S$

As earlier, we define a graph transformation as a set of parametrized operations. Each operation is a set $\Omega$ of parametrized guard-action pairs and a set of operation parameters. Formally, an operation $O$ is a pair $O = (\Omega, p)$ where $p$ is a vector of variables representing the operation parameters. An operation relates two states $s$ and $s'$ for a given set of operation parameters. A parameterized Guard-Action(GA) pair has as its parameters the operation parameters(inherited from the operation) and a set of GA parameters of its own. Intuitively, for a given set of parameters, the guard checks whether certain conditions are true of a state and if the guard holds true, actions are taken on the nodes and edges of the state. The action of each guard-action pair is described using a finite set of basic sub-actions $\kappa$. We allow non-determinism in the actions so that the GA pair can specify multiple sets of subactions on the nodes and edges of a state.

A Graph Transformation System $S$ is then defined as $S = (\Delta, \kappa)$ where $\Delta$ is a finite set of operations and $\kappa$ is the finite set of basic sub-actions in terms of which the action of any guard-action pair is expressed.

Mathematically, given a Graph Transformation System $S = (\Delta, \kappa)$, for an operation $O = (\Omega, p) \in \Delta$, a guard-action pair $\xi \in \Omega$ is defined as $\xi = (\Phi_G^\xi, \Phi_A^\xi)$. We look at each of these components of $\xi$.

1. $\Phi_G^\xi$:
   The guard is given by the formula $\Phi_G^\xi(f_\xi, p)$ which is an arbitrary FO formula, with free variables $f_\xi, p$, in the vocabulary $\sigma_s = \{N_s\} \cup \{E_s\} \cup \{T_{N,s}^i|1 \leq i \leq N_n\} \cup \{T_{E,s}^i|1 \leq i \leq N_e\}$

   $f_\xi$ denotes the GA parameters of $\xi$. Given values to $f_\xi, p$ then, the guard checks whether certain conditions are true in state $s$. (Note that $\sigma_s$ is the same as $\sigma_S$ except that each predicate of the former is subscripted with $s$)

2. $\Phi_A^\xi$:
   The action is given by the formula $\Phi_A^\xi(f_\xi, p)$. In order to define $\Phi_A^\xi(f_\xi, p)$, we first introduce predicates for the sub-actions of $\kappa$.

   The set $\kappa$ broadly consists of two kinds of sub-actions: (a) subactions on nodes $\kappa_N$ and (b) subactions on edges $\kappa_E$. (Thus $\kappa = \kappa_N \cup \kappa_E$).

   The set $\kappa_N$ consists of two kinds of sub-actions:
   - subactions $\kappa_{N,P} = \{\lambda_{P,1}, \lambda_{P,2}, \ldots, \lambda_{P,l}\}$ which affect the presence/absence of a node in going from state $s$ to state $s'$ and
   - subactions $\kappa_{N,T_i} = \{\lambda_{T_i,1}, \lambda_{T_i,2}, \ldots, \lambda_{T_i,l_i}\}$ ($1 \leq i \leq N_n$) which affect the presence/absence of type $i$ (in the set of types associated with a node) in going from state $s$ to state $s'$.
The set $\kappa_E$ consists of two kinds of sub-actions:

- subactions $\kappa_{E,P} = \{\mu_{P,1}, \mu_{P,2}, \ldots, \mu_{P,m}\}$ which affect the presence/absence of an edge in going from state $s$ to state $s'$ and
- subactions $\kappa_{E,T_i} = \{\mu_{T_{i,1}}, \mu_{T_{i,2}}, \ldots, \mu_{T_{i,m_i}}\}$ ($1 \leq i \leq N_e$) which affect the presence/absence of type $i$ (in the set of types associated with an edge) in going from state $s$ to state $s'$.

The guard, for its input parameters $(f_\xi, p)$, checks whether certain conditions are true of state $s$. If the guard evaluates to true, then the action labels each node and each edge with the sub-actions to be taken on it, for the parameters $(f_\xi, p)$.

We introduce the following set of predicates for the sub-actions. (Below, for a vector $f$, we denote the length of the vector as $|f|$.

- For each sub-action $\lambda_n \in \kappa_N$ we have the predicate $X_\xi^k(f, x, q)$ where $|f| = |f_\xi|$ and $|q| = |p|$. This predicate denotes that the action of $\xi$ labels node $x$ with the sub-action $\lambda_n$ for the parameters $(f, q)$.
- For each sub-action $\mu_e \in \kappa_E$ we have the predicate $Y_\xi^g(f, x, y, q)$ where $|f| = |f_\xi|$ and $|q| = |p|$. This predicate denotes that the action of $\xi$ labels edge $(x, y)$ with the sub-action $\mu_e$ for the parameters $(f, q)$.

For e.g.,

$$X_{P,1}^\xi(f_\xi, x, p) \land \bigwedge_{i=2}^{i=l} \neg X_{P,i}^\xi(f_\xi, x, p) \land \bigwedge_{j=1}^{j=N_t} X_{T,1}^\xi(f_\xi, x, p) \land \bigwedge_{j=1}^{j=N_e} \neg X_{T,1}^\xi(f_\xi, x, p)$$

means that the action of $\xi$ for parameters $f_\xi, p$ labels node $x$ with the subactions $\{\lambda_{P,1}\} \cup \{\lambda_{T,1}|1 \leq i \leq N_e\}$.

A set of sub-action labels specified by the action on all the nodes and edges for a set of parameter values of the GA pair forms a labelling for that set of parameter values. Thus a labelling produced by the action of $\xi$ for parameters $(f_\xi, p)$ is then the set of values of $X_\xi^k(f_\xi, x, p)$ and $Y_\xi^g(f_\xi, x, y, p)$ for all $x, y$ and for all $\lambda_n, \mu_e \in \kappa$. We allow non-determinism in the action so that the action can specify multiple labellings for the same set of parameters. If the guard evaluates to false, then no sub-action is labelled on any node or edge.

Let $\sigma_{N}^{a\xi} = \bigcup_{\xi \in \Omega} \{X_\xi^k|\lambda_n \in \kappa_N\}$; $\sigma_{E}^{a\xi} = \bigcup_{\xi \in \Omega} \{Y_\xi^g|\mu_e \in \kappa_E\}$

The action of the guard-action pair $\xi$ is then defined as $\Phi_{\xi}^A(f_\xi, p)$, an arbitrary FO formula, with free variables $f_\xi, p$, in the vocabulary

$$\sigma_0^A = \sigma_0 \cup \sigma_{N}^{a\xi} \cup \sigma_{E}^{a\xi}$$

with the constraint that for every occurrence of $X_\xi^k(f, x, q)$ and every occurrence of $Y_\xi^g(f, x, y, q)$ in the formula, $q$ is free and $q = p$. 
Note that the guard and the action have the same set of free variables, namely \( f_\xi \) and \( p \). Note also that by this definition, we allow non-determinism in the action of a guard-action pair so that the action can specify multiple labelings for a set of GA parameters.

We now define the labeling produced by a GA pair \( \xi \) for a given set of operation parameters \( p \). For each \( f_\xi \) value, if the guard evaluates to True, the corresponding action is taken. Thus, given a labeling produced by the action for each \( f_\xi \) value, the labeling produced by \( \xi \) is obtained by aggregating these labelings i.e. for each node(edge), the label produced on the node(edge) by \( \xi \) is obtained by taking the union of all the labels produced on the node(edge) by the action for all \( f_\xi \) values.

In case of non-determinism in the action, where for one set of GA parameter values, the action produces multiple labelings, we can see that multiple labelings are produced by the GA pair too.

We can similarly define the labeling produced by the operation for a given set of operation parameter values. Given a labeling produced by each GA pair of the operation, the labeling produced by the operation is obtained by aggregation i.e. for each node (edge), the label produced by the operation is obtained by taking the union of the labels produced on the node(edge) by all the GA pairs.

Again in case the GA pairs produce multiple labelings, we can see that multiple labelings are produced by the operation too.

To express the above, for each operation \( O = (\Omega, p) \in \Delta \), we introduce the following predicates:

For each node sub-action \( \lambda_n \in \kappa_N \), we introduce the predicate \( X_n^O(x, q)(|q| = |p|) \) which denotes that the operation \( O \) labels the node \( x \) with sub-action \( \lambda_n \) for parameters \( q \). Similarly for each edge sub-action \( \mu_e \in \kappa_E \), we introduce the predicate \( Y_e^O(x, y, q)(|q| = |p|) \) which denotes that the operation \( O \) labels the edge \((x, y)\) with sub-action \( \mu_e \) for parameters \( q \).

Then from the above, the operation \( O \) labels node \( x \) (edge \((x, y)\)) with \( \lambda_n \) (\( \mu_e \)) for parameters \( p \) iff the labeling produced by the action of some GA pair \( \xi \) for some GA parameter values \( f_\xi \) labels the node \( x \) with \( \lambda_n \) (edge \((x, y)\) with \( \mu_e \)). Thus we have

\[
\forall x \ X_n^O(x, p) \iff \bigvee_{\xi \in \Omega} \exists f X_n^\xi(f, x, p)
\]

\[
\forall x \forall y \ Y_e^O(x, y, p) \iff \bigvee_{\xi \in \Omega} \exists f Y_e^\xi(f, x, y, p)
\]

Denote the first formula as \( Z_n(p) \) and the second as \( Z_e(p) \). Let

\[
Z^O(p) = \bigwedge_{\lambda_n \in \kappa_N} Z_n(p) \land \bigwedge_{\mu_e \in \kappa_E} Z_e(p)
\]

A labeling produced by the operation for a given set of operation parameters is now translated to the next states of the operation for those set of parameters as follows.
5.3. DEFINING A GRAPH TRANSFORMATION SYSTEM $S$

Let $\sigma^O_N = \{X^O_n|\kappa_N \in \kappa_N\}$, $\sigma^O_E = \{Y^O_e|\kappa_e \in \kappa_E\}$

$\sigma_{s'} = \{N_{s'}\} \cup \{E_{s'}\} \cup \{T^i_{N,s'}|1 \leq i \leq N_n\} \cup \{T^i_{E,s'}|1 \leq i \leq N_e\}$

(Note that $\sigma_{s'}$ is the same as $\sigma_S$ except that each predicate of the former is subscripted with $s'$)

For each subaction $\lambda_n \in \kappa_N$, we have formula $\phi^O_n(x,p)$ defining the effect of the sub-action for operation $O$ i.e. $\phi^O_n(x,p)$ defines the condition that must be true of state $s'$ if $\lambda_n$ is taken on a node $x$ by operation $O$ for parameters $p$. Likewise we have formulae $\phi^O_e(x,y,p)$ defining the effect of subaction $\mu_e \in \kappa_E$ for $O$. We allow $\phi^O_n(x,p)$ (resp. $\phi^O_e(x,y,p)$) to be an arbitrary FO formula, with free variables $x,p$ (resp. $x,y,p$), in the vocabulary

$\sigma^O_{ss'} = \sigma_s \cup \sigma^O_N \cup \sigma^O_E \cup \sigma_{s'}$

with the constraint that for every occurrence of $X^O_n(x,q)$ and every occurrence of $Y^O_e(x,y,q)$ in the formula, $q$ is free and $q = p$.

Thus we have,

$$\forall x \bigwedge_{\lambda_n \in \kappa_N} (X^O_n(x,p) \rightarrow \phi^O_n(x,p)) \land$$

$$\forall x \forall y \bigwedge_{\mu_e \in \kappa_E} (Y^O_e(x,y,p) \rightarrow \phi^O_e(x,y,p))$$

Denote the above as $\Gamma^O_1(p)$.

For any node $x$, if no sub-action $\lambda_n \in \kappa_{N,P}$ is taken on it, its presence or absence is preserved in going from state $s$ to state $s'$. Similarly, if no sub-action $\lambda_n \in \kappa_{N,T_i}$ is taken on $x$, the presence or absence of type $i$ in the set of types of $x$ is preserved in going from $s$ to $s'$. Likewise for edges. Thus,

$$\bigwedge_{i=1}^{N_n} \forall x \left( \bigwedge_{\lambda_n \in \kappa_{N,T_i}} \neg X^O_n(x,p) \rightarrow (N_s(x) \leftrightarrow N_{s'}(x)) \land \right.$$

$$\bigwedge_{\lambda_n \in \kappa_{N,T_i}} \neg X^O_n(x,p) \rightarrow (T^i_{N,s}(x) \leftrightarrow T^i_{N,s'}(x)) \land$$

$$\forall x \forall y \left( \bigwedge_{\lambda_n \in \kappa_{N,T_i}} \neg X^O_n(x,y,p) \rightarrow (E_s(x,y) \leftrightarrow E_{s'}(x,y)) \land \right.$$

$$\bigwedge_{\lambda_n \in \kappa_{N,T_i}} \neg X^O_n(x,y,p) \rightarrow (T^j_{E,s}(x,y) \leftrightarrow T^j_{E,s'}(x,y)) \land$$

Denote the above by $\Gamma^O_2(p)$.

Let

$$\Gamma^O(p) = \Gamma^O_1(p) \land \Gamma^O_2(p)$$

Let

$$\Psi^\xi(f,p) = \left( \bigwedge_{\lambda_n \in \kappa_N} \neg X^\xi_n(f,x,p) \land \bigwedge_{\lambda_n \in \kappa_N} \neg Y^\xi_e(f,x,y,p) \right)$$
The transition relation for the guard-action pair \( \xi \), is then defined as

\[
T_{ss'}^{\xi} (p) = \left( \forall f_\xi : \Phi_G^\xi (f_\xi, p) \rightarrow \Phi_A^\xi (f_\xi, p) \land \right)
\]

\[
\forall f_\xi : \neg \Phi_G^\xi (f_\xi, p) \rightarrow \Psi_\xi (f_\xi, p)
\]

Thus, given a set of parameters \( p \), for all the values of \( f_\xi \) for which the condition specified by the guard \( \Phi_G^\xi (f_\xi, p) \) becomes true, the corresponding action \( \Phi_A^\xi (f_\xi, p) \) is true. For all the values of \( f_\xi \) for which the condition specified by \( \Phi_G^\xi (f_\xi, p) \) is false, no sub-action is taken on any node or edge for those values of \( f_\xi \).

We now see that given a state \( s \), putting together \( T_{ss'}^{\xi} (p) \) for all the GA pairs of operation \( O \) and \( Z^O(p) \), we get all the labellings produced by the operation \( O \). \( \Gamma^O(p) \) translates each labelling to a set of next states of the operation for parameters \( p \). Then putting all of these together, we get the transition relation \( T_{ss'}^O (p) \) of the operation. Thus

\[
T_{ss'}^O (p) = \left( \bigwedge_{\xi \in \Omega} T_{ss'}^{\xi} (p) \land \bigwedge \right)
\]

\[
Z^O(p) \land \Gamma^O(p)
\]

Finally the next states produced by the Graph Transformation System is obtained by taking the union of the next states produced by all the operations for all parameter values. Thus the transition relation \( T_{ss'} \) of the GTS \( S = (\Delta, \kappa) \) is given as

\[
T_{ss'} = \bigvee_{O=(\Omega, p) \in \Delta} \exists p \ T_{ss'}^O (p)
\]

Note that \( T_{ss'} \) is a sentence in the vocabulary

\[
\sigma_{ss'} = \bigcup_{O \in \Delta} \sigma_{ss'}^O
\]

5.4 Transition Relation on States Defined by \( S \)

We introduce the following notation:

For a vocabulary \( \sigma \) and \( \sigma_1 \subseteq \sigma \), given a \( \sigma \)-structure \( M \), the restriction of \( M \) to the predicates of \( \sigma_1 \) is denoted as \( M|_{\sigma_1} \).

As noted earlier \( \sigma_s \) and \( \sigma_{s'} \) are the same as \( \sigma_S \) except that the predicates of the former are subscripted with \( s \) and \( s' \) respectively. Then we can say that a \( \sigma_s \)-structure \( M \) defines a state \( s \) as follows:

- \( M \) and \( s \) have the same universe, \( A \) say.
5.4. TRANSITION RELATION ON STATES DEFINED BY $S$

- For all $x \in A$,
  
  $N_s(x) \iff N(x)$
  $T_{N,s}^i(x) \iff T_N^i(x) \quad 1 \leq i \leq N_n$

- For all $x, y \in A$,
  
  $E_s(x, y) \iff E(x, y)$
  $T_{E,s}^i(x, y) \iff T_E^i(x, y) \quad 1 \leq i \leq N_e$

(Likewise a $\sigma_{s'}$ structure also defines a state.)

Then a GTS $S$ defines a transition relation $T(s, s')$ on states as follows:

Given states $s$ and $s'$, $T(s, s')$ is true iff there exists a $\sigma_{ss'}$—structure $M$ such that $M|_{\sigma_s}$ defines $s$, $M|_{\sigma_{s'}}$ defines $s'$ and $M \models T_{ss'}$.

In the next chapter, we look at GTSes which use some special sub-actions. In all subsequent chapters, to simplify the notation, we shall omit the superscript $O$ in the notation when operation $O$ is clear from the context. Similarly we shall omit $\xi$ in the superscripts and subscripts when GA pair $\xi$ is clear from the context.
Chapter 6

Graph Transformation Systems With Special Sub-actions

Consider GTS $S = (\Delta, \rho)$ in which the sub-actions in $\rho$ are as follows:

- Sub-actions on nodes $\rho_N$:
  - Sub-actions for presence/absence of nodes, $\rho_{N,P}$
    1. $\lambda_{P,1} = \text{AddNode}$ - adds the parameter node which is absent in the graph of state $s$.
    2. $\lambda_{P,2} = \text{DelNode}$ - deletes the parameter node which is present in the graph of state $s$.
    3. $\lambda_{P,3} = \text{PresNode}$ - preserves (the presence or absence of) the parameter node.
  - Sub-actions for type-change of nodes, $\rho_{N,T_i}, 1 \leq i \leq N_n$
    1. $\lambda_{T_i,1} = \text{AddNodeType}_i$ - adds the type $i$ to the set of types of the parameter node where type $i$ is absent in the set of types of the parameter node in the state $s$.
    2. $\lambda_{T_i,2} = \text{DelNodeType}_i$ - deletes the type $i$ from the set of types of the parameter node where type $i$ is present in the set of types of the parameter node in the state $s$.
    3. $\lambda_{T_i,3} = \text{PresNodeType}_i$ - preserves (the presence or absence of) type $i$ in the set of types of the parameter node.

- Sub-actions on Edges $\rho_E$:
  - Sub-actions for presence/absence of edges, $\rho_{E,P}$
    1. $\mu_{P,1} = \text{AddEdge}$ - adds the parameter edge which is absent in the graph of state $s$. 

32
2. \( \mu_{P,2} = \text{DelEdge} \) - deletes the parameter edge which is present in the graph of state \( s \).

3. \( \mu_{P,3} = \text{PresEdge} \) - preserves (the presence or absence of) the parameter edge.

- **Sub-actions for type-change of edges, \( \rho_{E,T}, 1 \leq i \leq N_e \)**
  1. \( \mu_{T_i,1} = \text{AddEdgeType} \) - adds the type \( i \) to the set of types of the parameter edge where type \( i \) is absent in the set of types of the parameter edge in the state \( s \).
  2. \( \mu_{T_i,2} = \text{DelEdgeType} \) - deletes the type \( i \) from the set of types of the parameter edge where type \( i \) is present in the set of types of the parameter edge in the state \( s \).
  3. \( \mu_{T_i,3} = \text{PresEdgeType} \) - preserves (the presence or absence of) type \( i \) in the set of types of the parameter edge.

We can define the above sub-actions using the \( \sigma_s \) and the \( \sigma_{s'} \) predicates. For any operation \( O \in \Delta \), we can write the formulae \( \phi_n^O(\lambda_n \in \rho_N) \) and \( \phi_e^O(\mu_e \in \rho_E) \) as below.

We drop the \( n \) and \( e \) subscripts and the \( O \) superscript when these are clear from the context.

- **Formulae for sub-actions on nodes, \( \rho_N \):**
  - **Formulae for \( \rho_{N,P} \):**
    \[
    \lambda_{P,1} : \phi(x, p) = \alpha_{P,1}(x, p) = (\neg N_s(x) \land N_{s'}(x) \land p = p)
    \]
    \[
    \lambda_{P,2} : \phi(x, p) = \alpha_{P,2}(x, p) = (N_s(x) \land \neg N_{s'}(x) \land p = p)
    \]
    \[
    \lambda_{P,3} : \phi(x, p) = \alpha_{P,3}(x, p) = (N_s(x) \leftrightarrow N_{s'}(x) \land p = p)
    \]
  - **Formulae for \( \rho_{N,T}, (1 \leq i \leq N_t) \):**
    \[
    \lambda_{T_i,1} : \phi(x, p) = \alpha_{T_i,1}(x, p) = (\neg T_{N,s}^i(x) \land T_{N,s'}^i(x) \land p = p)
    \]
    \[
    \lambda_{T_i,2} : \phi(x, p) = \alpha_{T_i,2}(x, p) = (T_{N,s}^i(x) \land \neg T_{N,s'}^i(x) \land p = p)
    \]
    \[
    \lambda_{T_i,3} : \phi(x, p) = \alpha_{T_i,3}(x, p) = (T_{N,s}^i(x) \leftrightarrow T_{N,s'}^i(x) \land p = p)
    \]

- **Formulae for sub-actions on edges, \( \rho_E \):**
  - **Formulae for \( \rho_{E,P} \):**
    \[
    \mu_{P,1} : \phi(x, y, p) = \beta_{P,1}(x, p) = (\neg E_s(x, y) \land E_{s'}(x, y) \land p = p)
    \]
    \[
    \mu_{P,2} : \phi(x, y, p) = \beta_{P,2}(x, p) = (E_s(x, y) \land \neg E_{s'}(x, y) \land p = p)
    \]
    \[
    \mu_{P,3} : \phi(x, y, p) = \beta_{P,3}(x, p) = (E_s(x, y) \leftrightarrow E_{s'}(x, y) \land p = p)
    \]
- Formulae for $\rho_{E,T}(1 \leq i \leq N_e)$:
  
  \[
  \mu_{T_i,1} : \phi(x, y, p) = \beta_{T_i,1}(x, p) = (\neg T_{E,s}^i(x, y) \land T_{E,s'}^i(x, y) \land p = p)
  \]
  
  \[
  \mu_{T_i,2} : \phi(x, y, p) = \beta_{T_i,2}(x, p) = (T_{E,s}^i(x, y) \land \neg T_{E,s'}^i(x, y) \land p = p)
  \]
  
  \[
  \mu_{T_i,3} : \phi(x, y, p) = \beta_{T_i,3}(x, p) = (T_{E,s}^i(x, y) \leftrightarrow T_{E,s'}^i(x, y) \land p = p)
  \]

The special sub-actions defined above cover all the possibilities for a node and an edge -
(a) a node can either be added, deleted or preserved (b) type $i(1 \leq i \leq N_n)$ can either be added, deleted or preserved in the set of types of a node (c) an edge can either be added, deleted or preserved (d) type $i(1 \leq i \leq N_e)$ can either be added, deleted or preserved in the set of types of an edge.

Consider a GTS $S = (\Delta, \kappa)$. This defines a transition relation $T(s, s')$ on states. The question now is whether it is possible to construct a GTS $S' = (\Delta', \rho)$ such that $S'$ also defines the same transition relation $T(s, s')$ on states. In other words, is it the case that the transition relations on states defined by a general GTS can also be defined with a GTS that uses the special sub-actions $\rho$. Or is it that the former kind of GTS is more powerful than the latter kind in that the former enables us to capture transition relations on states which cannot be captured by the latter.

While this is an open question to be investigated, for the special kind of GTS described in the next chapter, the transition relation defined by it can indeed be defined by a GTS using the sub-actions $\rho$.

But before looking at this special kind of GTS, let's look at the GTS $S = (\Delta, \rho)$.

For any operation $O = (\Omega, p) \in \Delta_\rho$, we can write $\Gamma$ as

\[
\Gamma(p) = \Gamma_1(p) \land \Gamma_2(p)
\]

where $\Gamma_1(p)$ is given by

\[
\forall x \; \bigwedge_{\lambda_n \in \rho_N} (\chi_n(x, p) \rightarrow \alpha(x, p)) \land \forall x \forall y \; \bigwedge_{\mu_e \in \rho_E} (\chi_e(x, y, p) \rightarrow \beta(x, y, p))
\]

and $\Gamma_2(p)$ is given by

\[
\bigwedge_{i=1}^{N_n} \forall x \; \left( \bigwedge_{\lambda_n \in \rho_{N,P}} \neg \chi_n(x, p) \right) \rightarrow \alpha_{P,3}(x, p) \land \bigwedge_{i=1}^{N_e} \forall x \; \left( \bigwedge_{\lambda_n \in \rho_{N,T_i}} \neg \chi_n(x, p) \right) \rightarrow \alpha_{T_i,3}(x, p) \land \bigwedge_{i=1}^{N_e} \forall x \forall y \; \left( \bigwedge_{\mu_e \in \rho_{E,P}} \neg \chi_e(x, y, p) \right) \rightarrow \beta_{P,3}(x, y, p) \land \bigwedge_{i=1}^{N_e} \forall x \forall y \; \left( \bigwedge_{\mu_e \in \rho_{E,T_j}} \neg \chi_e(x, y, p) \right) \rightarrow \beta_{T_i,3}(x, y, p)
\]

Now,

\[
\alpha_{P,1}(x, p) \rightarrow \neg \alpha_{P,2}(x, p) \land \neg \alpha_{P,3}(x, p) \\
\rightarrow \neg \chi_{P,2}(x, p) \land \neg \chi_{P,3}(x, p) \land \neg (\bigwedge_{i=1}^{N_e} \neg \chi_{P,i}(x, p)) \\
\rightarrow \chi_{P,1}(x, p)
\]
Thus,

\[ \alpha_{P,1}(x, p) \leftrightarrow \chi_{P,1}(x, p) \]

Similarly we can show

\[ \alpha_{P,2}(x, p) \leftrightarrow \chi_{P,2}(x, p) \]
\[ \alpha_{P,3}(x, p) \leftrightarrow \bigwedge_{i=1}^{\infty} \neg \chi_{P,i}(x, p) \]

Then we can see that

\[ N_{s'}(x) \leftrightarrow \chi_{P,1}(x, p) \lor \left( N_{s}(x) \land \left( \bigwedge_{i=2}^{\infty} \neg \chi_{P,i}(x, p) \right) \right) \]

Now from Z, we have

\[ \bigwedge_{\lambda_n \in \rho_N} \forall x \chi_n(x, p) \leftrightarrow \bigvee_{\xi \in \Omega} \exists f \chi_\xi_n(f, x, p) \]

Then we can use the above equivalences to replace every occurrence of \( \chi_n \) by its corresponding equivalent expression in the RHS above. Thus we can write,

\[ N_{s'}(x) \leftrightarrow \left( \bigvee_{\xi \in \Omega} \exists f \chi_\xi_{P,1}(f, x, p) \right) \lor \left( N_{s}(x) \land \left( \bigwedge_{i=2}^{\infty} \neg \left( \bigvee_{\xi \in \Omega} \exists f \chi_\xi_{P,i}(f, x, p) \right) \right) \right) \]

Call the RHS above as \( \varsigma_{N,P}(x, p) \).

Similarly if

\[ \varsigma_{E,P}(x, y, p) = \left( \bigvee_{\xi \in \Omega} \exists f \Upsilon_{P,1}^\xi(f, x, y, p) \right) \lor \left( E_{s}(x, y) \land \left( \bigwedge_{j=2}^{\infty} \neg \left( \bigvee_{\xi \in \Omega} \exists f \Upsilon_{P,j}^\xi(f, x, y, p) \right) \right) \right) \]
\[ \varsigma_{N,T_i}(x, y, p) = \left( \bigvee_{\xi \in \Omega} \exists f \chi_{T_{i,1}}^\xi(f, x, y, p) \right) \lor \left( T_{N,s}^j(x, y) \land \left( \bigwedge_{j=2}^{\infty} \neg \left( \bigvee_{\xi \in \Omega} \exists f \chi_{T_{i,j}}^\xi(f, x, y, p) \right) \right) \right) \]
\[ , 1 \leq i \leq N_n \]
\[ \varsigma_{E,T_i}(x, y, p) = \left( \bigvee_{\xi \in \Omega} \exists f \Upsilon_{T_{i,1}}^\xi(f, x, y, p) \right) \lor \left( T_{E,s}^j(x, y) \land \left( \bigwedge_{j=2}^{\infty} \neg \left( \bigvee_{\xi \in \Omega} \exists f \Upsilon_{T_{i,j}}^\xi(f, x, y, p) \right) \right) \right) \]
\[ , 1 \leq i \leq N_e \]

then, we can see that for any operation \( O \in \Delta \),

\[ E_{s'}(x, y) \leftrightarrow \varsigma_{E,P}(x, y, p) \]
\[ T_{N,s'}^j(x, y) \leftrightarrow \varsigma_{N,T_i}(x, y, p) \]
\[ 1 \leq i \leq N_n \]
\[ T_{E,s'}^j(x, y) \leftrightarrow \varsigma_{E,T_i}(x, y, p) \]
\[ 1 \leq i \leq N_e \]
The above expressions will help us in the next chapter. We will also need the following result in the next chapter.

**Claim 1:**

Let $\sigma$ be a relational vocabulary and $\sigma_1 \subseteq \sigma$. Let $\varphi(f)$ be a formula over $\sigma_1$. Let $M$ be a $\sigma$-structure with universe $A$. Then for any $f_1 \in A^{|f|}$,

$$M \models \varphi(f) \iff M|_{\sigma_1} \models \varphi(f_1)$$

**Proof:**

Since $\varphi(f)$ is a formula over $\sigma_1$, the truth of $\varphi(f_1)$ is dependent only on the interpretations of the predicates of $\sigma_1$ and is independent of the interpretations of the predicates of $\sigma - \sigma_1$. 

\qed
Chapter 7

A Special Kind of Graph Transformation Systems

Firstly we recall that for any GTS, for any operation \( O = (\Omega, p) \),

\[
Z^O(p) = \left( \bigwedge_{\lambda_n \in \kappa_N} \forall x \bigvee \lambda_n \in \left[ X^O_n(x, p) \leftrightarrow \bigvee_{\xi \in \Omega} \exists f X^\xi_n(f, x, p) \wedge \bigvee_{\lambda_n \in \kappa_N} \forall x \forall y Y^O_e(x, y, p) \leftrightarrow \bigvee_{\xi \in \Omega} \exists f Y^\xi_e(f, x, y, p) \right) \right)
\]

Using the equivalences, we can eliminate the \( X^O_n \) and the \( Y^O_e \) predicates by replacing every occurrence of \( X^O_n \) and \( Y^O_e \) in \( Z^O \) and \( O \) by their corresponding equivalent expressions to get

\[
C^O(p) = \Gamma^O \left[ \left( \bigvee_{\xi \in \Omega} \exists f X^\xi_n(f, x, p) \right) | X^O_n(x, p) \right] \left[ \left( \bigvee_{\xi \in \Omega} \exists f Y^\xi_e(f, x, y, p) \right) | Y^O_e(x, y, p) \right]
\]

In the above \([a|b]\) denotes the replacement of every occurrence of \( b \) with \( a \). Then we can write \( T^O_{ss'}(p) \) as

\[
T^O_{ss'}(p) = \bigwedge_{\xi \in \Omega} T^\xi_{ss'}(p) \wedge C^O(p)
\]

Consider now a GTS \( S = (\Delta, \kappa) \) in which for each operation \( O = (\Omega, p) \in \Delta \), we have the property as described below:

For each \( \xi \in \Omega \), for each \( \lambda_n \in \kappa_N \) and each \( \mu_e \in \kappa_E \), there exist formulae \( \varphi^\xi_n(f, x, p) \) and \( \varphi^\xi_e(f, x, y, p) \) respectively in the vocabulary

\[
\sigma_s \cup \{ \lambda_m \mid \lambda_m \in \rho_N, \xi \in \Omega \} \cup \{ \gamma^\xi_f \mid \mu_f \in \rho_E, \xi \in \Omega \}
\]
\( \varphi_n^x \) and \( \varphi_e^x \) are such that every occurrence of \( \chi_n^x \) (resp. \( \Upsilon_f^x \)) is of the form \( \chi_n^x(f_1, x, q) \) (resp. \( \Upsilon_f^x(f_1, x, y, q) \)) where (a) \( |f_1| = |f_e| \) and (b) \( q \) is free and \( q = p \).

Then the following property is true.

\[
\bigwedge_{x \in \Omega} \left( \forall f \forall x \bigwedge_{\lambda_n \in \kappa_N} X_n^x(f, x, p) \iff \varphi_n^x(f, x, p) \wedge \right)
\]

Call the above as \( C_1(p) \). Let \( C_2(p) = \)

\[
\bigwedge_{\lambda_n \in \rho_N} \forall x \left( \forall \xi \in \Omega \exists f \chi_n^x(f, x, p) \right) \rightarrow \beta_n(x, p) \wedge \\
\bigwedge_{\mu_f \in \rho_E} \forall x \forall y \left( \forall \xi \in \Omega \exists f \Upsilon_e^x(f, x, p) \right) \rightarrow \beta_e(x, y, p) \wedge \\
\bigwedge_{i=1}^{N_n} \forall x \left( \forall \lambda_n \in \rho_N, p \rightarrow \left( \forall \xi \in \Omega \exists f \chi_n^x(f, x, p) \right) \right) \rightarrow \beta_{N, p, 3}(x, p) \wedge \\
\bigwedge_{i=1}^{N_e} \forall x \forall y \left( \forall \lambda_n \in \rho_N, T_i \rightarrow \left( \forall \xi \in \Omega \exists f \Upsilon_e^x(f, x, y, p) \right) \right) \rightarrow \beta_{N, T, 3}(x, p) \wedge \\
\bigwedge_{i=1}^{N_e} \forall x \forall y \left( \forall \lambda_n \in \rho_N, T_j \rightarrow \left( \forall \xi \in \Omega \exists f \Upsilon_e^x(f, x, y, p) \right) \right) \rightarrow \beta_{E, T, 3}(x, y, p)
\]

(\text{Note the similarity between } \text{\textit{C}_2}(p) \text{ and } C^0(p).)

Let \( X \) denote the vector of all the \( X_n^x \) predicates (\( \lambda_n \in \kappa_N, \xi \in \Omega \)). Similarly let \( Y, \chi \) and \( \Upsilon \) denote the vector of all \( Y_e^x, \chi_n^x \) and the \( \Upsilon_f^x \) predicates.

Then the condition satisfied by the GTS for each operation \( O \) is

\[
\exists p T^0_{ss}(p) \rightarrow \exists p \exists X Y \chi \Upsilon \left( \begin{array}{c}
T^0_{ss}(p)(X, Y) \\
C_1(p)(X, Y, \chi, \Upsilon) \wedge \\
C_2(p)(\chi, \Upsilon)
\end{array} \right)
\]

(The above is thus a second-order condition.)

We now construct a GTS \( S' = (\Delta', \rho) \) such that for each operation \( O = (\Omega, p) \in \Delta \), we have \( O = (\Omega', p) \in \Delta' \) as follows:

- Let \( \Omega = \{\xi_1, \xi_2, \ldots, \xi_r\} \). Then \( \Omega' = \{\xi'_1, \xi'_2, \ldots, \xi'_r\} \)
- \( \xi' = (\Phi_G, \Phi_A) \) where
  - For \( \xi' \in \{\xi'_2, \ldots, \xi'_r\} \)
    \( \Phi_G(f_{\xi'}, p) = \Phi_A(f_{\xi'}, p) = \text{True} \)
    \( |f_{\xi'}| = |f_{\xi}| \)
For $\xi' = \xi_1$,  
$\Phi_G(f_{\xi'}, p) = \text{True}$  
$|f_{\xi'}| = |f_{\xi}|$  
$\Phi_A(f_{\xi'}, p)$ is defined as follows:
Let  
$\varphi_n^\xi(f, x, p) = \varphi_n^\xi(f, x, p)[\xi'|\xi]$  
$\varphi_n^\xi(f, x, y, p) = \varphi_n^\xi(f, x, y, p)[\xi'|\xi]$  
$C_2'(p) = C_2(p)[\xi'|\xi]$  
$\zeta_{N,P}'(x, p) = \zeta_{N,P}(x, p)[\xi'|\xi]$  
$\zeta_{E,P}'(x, y, p) = \zeta_{E,P}(x, y, p)[\xi'|\xi]$  
$\zeta_{N,T_i}'(x, p) = \zeta_{N,T_i}(x, p)[\xi'|\xi]$  
$\zeta_{E,T_i}'(x, y, p) = \zeta_{E,T_i}(x, y, p)[\xi'|\xi]$  
Then,  
$\Phi_A(f_{\xi'}, p) = (f_{\xi'} = f_{\xi'}) \land$  
$T_{ss'}^{O'}(p) \left[ \varphi_n^\xi[X_n, \varphi_e^\xi[Y_e] \at \zeta_{N,P}'(x, p)|N_s(x) \at \zeta_{E,P}'(x, y, p)|E_s'(x, y) \at \zeta_{N,T_i}'(x, p)|T_{N,s_i}(x), 1 \leq i \leq N_n \at \zeta_{E,T_i}'(x, y, p)|T_{E,s_i}(x, y), 1 \leq i \leq N_e \right]$  

The transition relation, $T_{ss'}^{O'}$ is given by  
$T_{ss'}^{O'}(p) = \bigwedge_{\xi' \in \Omega'} T_{ss'}^{\xi'}(p) \land C^{O'}(p)$  
$= \left( \forall f_{\xi'} \Phi_A^{\xi'}(f_{\xi'}, p) \right) \land C^{O'}(p)$  

Note that,  
$C^{O'}(p) = C_2'(p)$  

Now, from $C^{O'}$, as seen in the previous chapter,  
$N_{s'}(x) \leftrightarrow \zeta_{N,P}'(x, p)$  
$E_{s'}(x, y) \leftrightarrow \zeta_{E,P}'(x, y, p)$  
$T_{N,s_i}'(x) \leftrightarrow \zeta_{N,T_i}'(x, p), 1 \leq i \leq N_n$  
$T_{E,s_i}'(x, y) \leftrightarrow \zeta_{E,T_i}'(x, y, p), 1 \leq i \leq N_e$  

Using these equivalences, we can replace every occurrence of $\zeta_{N,P}', \zeta_{E,P}', \zeta_{N,T_i}', \zeta_{E,T_i}'$ with their corresponding LHS above, in $T_{ss'}^{O'}$ to get
\[ T_{ss'}^O(p) = T_{ss'}^O(p) \left[ \varphi_n | X_n^\xi, \varphi_e | Y_e^\xi \right] \land C_2'(p) \]

We now prove that

1. If there is a \( \sigma_{ss'}^O \)-structure \( M \) such that \( M \models \exists p T_{ss'}^O(p) \), then there is a \( \sigma_{ss'}^{O'} \)-structure \( M' \) such that \( M' \models \exists p T_{ss'}^{O'}(p) \) and \( M|_{\sigma_{ss'}^O \cup \sigma_{ss'}^{O'}} = M'|_{\sigma_{ss'}^O \cup \sigma_{ss'}^{O'}} \)

2. If there is a \( \sigma_{ss'}^{O'} \)-structure \( M \) such that \( M \models \exists p T_{ss'}^{O'}(p) \), then there is a \( \sigma_{ss'}^O \)-structure \( M' \) such that \( M' \models \exists p T_{ss'}^O(p) \) and \( M|_{\sigma_{ss'}^O \cup \sigma_{ss'}^{O'}} = M'|_{\sigma_{ss'}^O \cup \sigma_{ss'}^{O'}} \)

Part 1:

Let the universe of \( M \) be \( A \). By the condition satisfied by the GTS, \( S \), there exists \( p_i \in A^{[p]} \) and interpretations \( X_1, Y_1, \chi_1, \Psi_1 \) such that

\[
M \models \left( \begin{array}{c}
T_{ss'}^O(p)(X_1, Y_1) \land C_1(p)(X_1, Y_1, \chi_1, \Psi_1) \land C_2(p)(\chi_1, \Psi_1)
\end{array} \right)
\]

Let \( \sigma_1 = \sigma_{ss'}^O \cup \sigma_{ss'}^{O'} \). Construct the following \( \sigma_1 \)-structure, \( M_1 : \)

1. It has the same universe as \( M \), i.e. \( A \).

2. \( M_1|_{\sigma_{ss'}^O \cup \sigma_{ss'}^{O'}} = M|_{\sigma_{ss'}^O \cup \sigma_{ss'}^{O'}} \)

3. The interpretations of the \( X_n^\xi, Y_e^\xi, \chi_m^\xi \) and the \( \Psi_j^\xi \) predicates is as follows:

\[
\begin{array}{lcl}
X_n^\xi(f, x, q) & \leftrightarrow & X_{1n}(f, x, q) \\\nY_e^\xi(f, x, y, q) & \leftrightarrow & Y_{1e}(f, x, y, q) \\\n\chi_m^\xi(f, x, q) & \leftrightarrow & \chi_{1m}(f, x, q) \\\n\Psi_j^\xi(f, x, y, q) & \leftrightarrow & \Psi_{1j}(f, x, y, q)
\end{array}
\]

Above \( f \in A^{[f]}, q \in A^{[q]}, x \in A, \lambda_n \in \kappa_N, \mu_e \in \kappa_e, \lambda_m \in \rho_N, \mu_f \in \rho_E, \xi \in \Omega \). Also \( X_{1n}^\xi \) is the interpretation of the \( X_n^\xi \) predicate of \( X_1 \). Similarly for \( Y_{1e}, \chi_{1m}^\xi \) and \( \Psi_{1j}^\xi \).

Let \( C_1'(p) = C_1(p) \left[ \varphi_n | \varphi_n^\xi, \varphi_e^\xi | \varphi_e^\xi \right] \)

Then we can see that

\[
M_1 \models (T_{ss'}^O(p_1) \land C_1'(p_1) \land C_2'(p_1))
\]
Using the equivalences in \( C_1'(p_1) \), we can write

\[
M_1 \models (T_{ss'}(p_1) \land C_1'(p_1) \land C_2'(p_1)) [\varphi^\xi_n | X^\xi_n, \varphi^\xi_e | Y^\xi_e] \\
\models (T_{ss'}(p_1) [\varphi^\xi_n | X^\xi_n, \varphi^\xi_e | Y^\xi_e] \land \text{True} \land C_2'(p_1)) \\
\models T_{ss'}(p_1) \\
\models \exists p T_{ss'}(p)
\]

Now the formula \( \exists p T_{ss'}(p) \) is over the vocabulary \( \sigma_{ss'} \subseteq \sigma_1 \). Then using Claim 1, we have

\[
M_1 |_{\sigma_{ss'}} \models \exists p T_{ss'}(p)
\]

Taking \( M' = M_1 |_{\sigma_{ss'}} \), we find that \( M' \) is a \( \sigma_{ss'} \)-structure such that \( M'|_{\sigma_1 \cup \sigma_{ss'}} = M |_{\sigma_1 \cup \sigma_{ss'}} \)

Part 2:

Consider a \( \sigma_{ss'} \)-structure \( M \) with universe \( A \) such that

\[
M \models \exists p T_{ss'}(p)
\]

Then there exists \( p_1 \in A^{|p|} \) such that

\[
M \models T_{ss'}(p_1)
\]

Extend \( M \) to a \( \sigma_1 \)-structure \( M_1 \) by using the following definitions for the \( X^\xi_n \) and the \( Y^\xi_e \) predicates.

\[
X^\xi_n(f, x, q) \iff \varphi^\xi_n(f, x, q) \\
Y^\xi_e(f, x, y, q) \iff \varphi^\xi_e(f, x, y, q)
\]

Above, \( f \in A^{|\xi|}, q \in A^{|p|}, x \in A, \lambda_n \in \kappa_N, \mu_e \in \kappa_E, \xi \in \Omega \).

Then

\[
M_1 \models T_{ss'}(p_1) \land C_1'(p_1)
\]

Using the equivalences in \( C_1'(p_1) \), we can write

\[
M_1 \models (T_{ss'}(p_1) \land C_1'(p_1)) [X^\xi_n | \varphi^\xi_n, Y^\xi_e | \varphi^\xi_e] \\
\models (T_{ss'}(p_1) [\varphi^\xi_n | X^\xi_n, \varphi^\xi_e | Y^\xi_e] \land C_2'(p_1)) \land C_1'(p_1) \\
\models (T_{ss'}(p_1) \land C_2'(p_1) \land \text{True}) \\
\models T_{ss'}(p_1) \\
\models \exists p T_{ss'}(p)
\]
Now the formula $\exists p T^{O}_{ss'}(p)$ is over the vocabulary $\sigma^{O}_{ss'} \subseteq \sigma_1$. Then using Claim 1, we have

$$M_{1}|_{\sigma^{O}_{ss'}} \models \exists p T^{O}_{ss'}(p)$$

Taking $M' = M_{1}|_{\sigma^{O}_{ss'}}$, we find that $M'$ is a $\sigma^{O}_{ss'}$-structure such that $M'|_{\sigma_1 \cup \sigma_f} = M|_{\sigma_1 \cup \sigma_f}$.

Thus $S' = (\Delta', \rho)$ defines the same transition relation on states as $S = (\Delta, \kappa)$. 

$\square$
Chapter 8

Bounded Model Property

In chapter 5, from the logical formalism for GTSes, the transition relation for the GTS was expressed by $T_{s_0'}$, which is a first order (FO) logic formula. We consider safety properties and initial state conditions which also are FO expressible properties of states. Thus for a state $s$

- $\mathcal{P}(s)$ is true iff $s \models \varphi_P$ where $\varphi_P$ is a $\sigma_S$—sentence expressing the safety property.
- $\mathcal{I}(s)$ is true iff $s \models \varphi_I$ where $\varphi_I$ is a $\sigma_S$—sentence expressing the initial-state conditions.

We recall that the procedure \textbf{InductionSolve} uses a validity checker which is used to check for the validity of the formulae

$$\mathcal{I}(s_1) \rightarrow \mathcal{P}(s_1)$$
$$\mathcal{P}(s_1)$$

and for $k \geq 1$

$$\left( \bigwedge_{i=1}^{i=k} T(s_i, s_{i+1}) \land \bigwedge_{i=1}^{i=k} \mathcal{P}(s_i) \land \bigwedge_{i=2}^{i=k+1} \neg \mathcal{I}(s_i) \land \bigwedge_{1 \leq i,j \leq k; i \neq j} (s_i \neq s_j) \land \mathcal{I}(s_1) \right) \rightarrow \mathcal{P}(s_{k+1})$$

$$\left( \bigwedge_{i=1}^{i=k} T(s_i, s_{i+1}) \land \bigwedge_{i=1}^{i=k} \mathcal{P}(s_i) \land \bigwedge_{1 \leq i,j \leq k; i \neq j} (s_i \neq s_j) \right) \rightarrow \mathcal{P}(s_{k+1})$$

Since $T_{s_0'}, \varphi_P$ and $\varphi_I$ are FO formulae, the validity checker then checks the validity of a FO formula. However, FO is known to be undecidable in general. Hence, no validity checker can be guaranteed to terminate for arbitrary $T_{s_0'}, \varphi_P$ and $\varphi_I$. The non-termination of the validity checker causes the non-termination of \textbf{InductionSolve}. We also saw that
we cannot fix a bound on the iterator $k$ in \texttt{InductionSolve} and expect the latter would terminate within the bound and give the correct output. The unbounded increase of the iterator then causes non-termination of \texttt{InductionSolve}. Thus there are two facets to the non-termination of \texttt{InductionSolve}:

1. Non-termination due to unbounded increase of the iterator.
2. Non-termination of the validity checker.

While (1) above cannot be avoided (whether or not the validity checker terminates) in general, we can avoid (2) by considering special classes of graph transformation systems, safety properties and initial state conditions for which we can construct a validity checker that would decide the validity of the formulae appearing in \texttt{InductionSolve} mentioned above.

We now observe the following:

Checking the validity of a formula is the same as checking the satisfiability of the negation of the formula. Hence considering the negation of the above formula, we have

$$ IS1(0) = \mathcal{I}(s_1) \land \neg \mathcal{P}(s_1) $$
$$ IS2(0) = \neg \mathcal{P}(s_1) $$

and for $k \geq 1$

$$ IS1(k) = \bigwedge_{i=1}^{i=k} T(s_i,s_{i+1}) \land \bigwedge_{i=1}^{i=k} \mathcal{P}(s_i) \land \bigwedge_{i=2}^{i=k+1} \neg \mathcal{I}(s_i) \land \bigwedge_{1 \leq i,j \leq k; i \neq j} (s_i \neq s_j) \land \mathcal{I}(s_1) \land \neg \mathcal{P}(s_{k+1}) $$
$$ IS2(k) = \bigwedge_{i=1}^{i=k} T(s_i,s_{i+1}) \land \bigwedge_{i=1}^{i=k} \mathcal{P}(s_i) \land \bigwedge_{1 \leq i,j \leq k; i \neq j} (s_i \neq s_j) \land \neg \mathcal{P}(s_{k+1}) $$

We then wish to check whether the above formulae are satisfiable. Consider $IS1(k)$ above. If it is satisfiable, there exists a sequence of states $s_1, s_2, \ldots, s_{k+1}$ satisfying $IS1(k)$. Suppose that for every such sequence, we can find a sequence $s'_1, s'_2, \ldots, s'_{k+1}$ satisfying $IS1(k)$ such that each $s'_i$ is bounded in size (i.e. has a bounded number of nodes and edges). If we know the bounds $b_i$ on the sizes of each of the states $s'_i$, then there is only a finite number of sequences $s'_1, s'_2, \ldots, s'_{k+1}$ of states such that each $s'_i$ is of size at most $b_i$. We can exhaustively search these sequences and check if we can find any sequence which satisfies $IS1(k)$. If we find one, we know that $IS1(k)$ is satisfiable and we also would have found a satisfying assignment for $s_1, s_2, \ldots, s_{k+1}$ (namely $s'_1, s'_2, \ldots, s'_{k+1}$). Else we know that there cannot exist any sequence of states $s_1, s_2, \ldots, s_{k+1}$ satisfying $IS1(k)$ because if there was, we should have been able to find a sequence $s'_1, s'_2, \ldots, s'_{k+1}$ (in which the size of $s'_i$ is bounded by $b_i$) corresponding to it and satisfying $IS1(k)$. Thus the bounded model
property of IS1(k) by which if it has a model, it also has a model of bounded size enables us to decide the satisfiability of IS1(k). Similarly we can decide satisfiability of the other formulae if they also have the bounded model property.

We then investigate which graph transformation systems, initial state conditions and safety properties have the bounded model property, so that we can use this to get bounded models for the formulae IS1(k) and IS2(k) above.

In the forthcoming chapter, we shall look at 2 kinds of GTSes which have bounded models for Tss. We later also look some safety properties and initial state conditions which have bounded models. In this section, we look at two results that will help in proving existence of bounded models for the GTSes we shall consider.

We introduce the following notation:

If M1 is a substructure of M, we denote it as M1 ⊆ M.

**Claim 2:**

Let ϕ(f) be a quantifier-free formula over the vocabulary σ. Let M be a σ-structure with universe A. Then for each f1 ∈ A|f|, for each A1 ⊆ A such that f1 ∈ A1|f|, the substructure M1 of M generated by A1 satisfies

\[ M_1 \models \varphi(f_1) \iff M \models \varphi(f_1) \]

**Proof:**

Let f = (f1, f2, ..., fn). For each f1 = (f11, f12, ..., f1n) ∈ A|f|, if f1 ∈ A1|f|, A1 ⊆ A, then in the substructure M1 of M generated by A1, we have

\[ M_1 \models P(f_{i_11}, f_{i_22}, ..., f_{i_kk}) \iff M \models P(f_{i_11}, f_{i_22}, ..., f_{i_kk}), \quad P \in \sigma, P \text{ - ary} \]

\[ 1 \leq i_j \leq n, \quad 1 \leq j \leq k \]

\[ M_1 \models (f_{i_1i_1} = f_{i_2i_2}) \iff M \models (f_{i_1i_1} = f_{i_2i_2}) \quad 1 \leq i_1, i_2 \leq n \]

Since ϕ(f1) is a quantifier-free formula, it is a boolean combination of the above atomic formulae. Hence

\[ M_1 \models \varphi(f_1) \iff M \models \varphi(f_1) \]
Claim 3:

Let \( \varphi(f) \) be a \( \exists^* \forall^* \) formula (in prenex-normal form) over vocabulary \( \sigma \) with \( m \) existential quantifiers. Let \( M \) be a \( \sigma \)-structure with universe \( A \) such that \( M \models \varphi(f_1) \). Then there exists \( A_1 \subseteq A \) such that

1. \( f_1 \in A_1^n \)
2. \(|A_1| \leq ||f_1|| + m\)
3. \( \forall A_2, A_1 \subseteq A_2 \subseteq A \), the substructure \( M_2 \) of \( M \) generated by \( A_2 \) satisfies \( M_2 \models \varphi(f) \)

(Above, \( ||f_1|| \) denotes the number of distinct elements in \( f_1 \).)

Proof:

- Case 1: \( m = 0 \)
  Then \( \varphi(f) = \forall y_1, y_2, \ldots, y_n \varphi_1(y_1, y_2, \ldots, y_n, f) \), \( n \geq 0 \). Let \( A_1 \) be such that \( A_1 \) contains only the distinct elements of \( f_1 \). Then \( f_1 \in A_1^n, |A_1| \leq ||f_1|| \). Consider \( A_2 \) such that \( A_1 \subseteq A_2 \subseteq A \). Let \( M_2 \) be the substructure of \( M \) generated by \( A_2 \).

  Suppose \( M_2 \models \neg \varphi(f_1) \)
  i.e. \( M_2 \models \neg \forall y_1, y_2, \ldots, y_n \varphi_1(y_1, y_2, \ldots, y_n, f_1) \)
  Then there exist \( y_{11}, y_{12}, \ldots, y_{1n} \in A_2 \) such that

  \[
  M_2 \models \neg \varphi_1(y_{11}, y_{12}, \ldots, y_{1n}, f_1)
  \]

  Since \( \varphi_1 \) is a quantifier free formula and \( M_2 \) is a substructure of \( M \), by Claim 2,

  \[
  M \models \neg \varphi_1(y_{11}, y_{12}, \ldots, y_{1n}, f_1) \\
  \models \exists y_1, y_2, \ldots, y_n \neg \varphi_1(y_1, y_2, \ldots, y_n, f_1) \\
  \models \neg \varphi(f_1)
  \]

  which is a contradiction. Hence \( M_2 \models \varphi(f_1) \).

- Case 2: \( m > 0 \)
  Then \( \varphi(f) = \exists x_1, x_2, \ldots, x_m \forall y_1, y_2, \ldots, y_n \varphi_1(x_1, x_2, \ldots, x_m, y_1, y_2, \ldots, y_n, f) \), \( n \geq 0 \).
  As \( M \models \varphi(f_1) \), there are elements \( x_{11}, x_{12}, \ldots, x_{1m} \in A \) such that

  \[
  M \models \forall y_1, y_2, \ldots, y_n \varphi_1(x_{11}, x_{12}, \ldots, x_{1m}, y_1, y_2, \ldots, y_n, f_1)
  \]
As the above is a $\forall^*$ formula, by Case 1, there exists $A_1 \subseteq A$ such that $f_i \in A_1^{f_i}$, $x_i \in A_1(1 \leq i \leq m)$, $|A_1| \leq ||f_i|| + m$ and $\forall A_2, A_1 \subseteq A_2 \subseteq A$, the substructure $M_2$ of $M$ generated by $A_2$ is such that

$$M_2 \models \forall y_1, y_2, \ldots, y_n \varphi_1(x_{11}, x_{12}, \ldots, x_{1m}, y_1, y_2, \ldots, y_n, f_1)$$

$$\models \exists x_1, x_2, \ldots, x_m, \forall y_1, y_2, \ldots, y_n \varphi_1(x_1, x_2, \ldots, x_m, y_1, y_2, \ldots, y_n, f_1)$$

$$\models \varphi(f_1)$$
Chapter 9

Graph Transformation Systems

Having Bounded Models

9.1 Class 1

9.1.1 An Example - The Library System

We now express the library system as a Graph Transformation System. Firstly we note that for the library system

\[ N_t = \{ \text{Book}(B), \text{Member}(M), \text{Claim}(C), \text{Loan}(L), \text{Shelf}(S) \} \]

Then \( T^1_N(x) = B(x), T^2_N(x) = M(x), T^3_N(x) = C(x), T^4_N(x) = L(x), T^5_N(x) = S(x) \). There are no types for the edges so \( E_t = \{ \} \). (Since there are no types for the edges, there are no type change sub-actions in \( \rho \).) The operations in the GTS for the library system correspond to the operations of \( \text{Addbook}, \text{Addmember}, \text{Reserve}, \text{Borrow}, \) and \( \text{Return} \). We shall first look at the operation for \( \text{Reserve} \).

\( \text{Reserve}(t,m) \):

1. If at least one book with the title \( t \) is on shelf, then exactly one such book is removed from the shelf and reserved for the member \( m \). As a graph transformation, if there is a \( B \) node adjacent to the \( T \) node \( t \) and adjacent to an \( S \) node and there exists a \( C \) node which is absent in the graph of the current state, then for exactly one \( B \) node \( b \), one \( S \) node \( s \) and one \( C \) node \( c \) satisfying the properties just mentioned, the edges between \( b \) and \( s \) is deleted, \( c \) is added, and the edge between \( b \) and \( c \) and between \( c \) and \( m \) are added. Other nodes and edges are preserved.

2. If there is no book on the shelf with title \( t \), the claim is kept pending. As a graph transformation, if there is no \( B \) node \( b \) and \( S \) node \( s \) such that \( b \) and \( t \) are adjacent
and \( b \) and \( s \) are adjacent and there exists a \( C \) node which is absent in the graph of the current state, then exactly one \( C \) node \( c \) which is absent in the graph of the current state is added and the \( c - t \) and the \( c - m \) edges are added. Other nodes and edges are preserved.

We can construct an operation \( O_{\text{Reserve}} = (\{\xi_1, \xi_2\}, p) \) where \( p = (t, m) \) as follows: 
\[
\xi_1 = (\Phi_G^f, \Phi_A^f), \ f_t = (b, s, c).
\]
This corresponds to (1) above.

(Henceforth, when \( \xi \) is clear from the context, we shall omit it in the subscript and superscript in the notation.). Thus

\[
\xi_1 = (\Phi_G, \Phi_A), \ f = (b, s, c)
\]

\[
\begin{align*}
\Phi_G(f, p) &= M_s(m) \wedge T_s(t) \wedge B_s(b) \wedge S_s(s) \wedge C_s(c) \wedge \\
& \quad \quad N_s(m) \wedge N_s(t) \wedge N_s(b) \wedge N_s(s) \wedge \neg N_s(c) \wedge \\
& \quad \quad E_s(b, t) \wedge E_s(b, s) \\
\Phi_A(f, p) &= \exists f_1 ((f_1 \neq f) \wedge \psi(f_1, p)) \rightarrow \Psi(f, p) \wedge \\
& \quad \quad \neg \exists f_1 ((f_1 \neq f) \wedge \psi(f_1, p)) \rightarrow \psi(f, p) \\
\psi(f, p) &= \chi_{P,1}(f, c, p) \wedge \\
& \quad \quad \chi_{P,2}(f, b, c, p) \wedge \chi_{P,1}(f, c, m, p) \wedge \chi_{P,2}(f, b, s, p) \wedge \\
& \quad \quad \forall x ((x \neq c) \rightarrow \chi_{P,3}(f, x, p)) \wedge \\
& \quad \quad \forall x, y ((x = b \wedge y = s) \lor (x = b \wedge y = c) \lor (x = c \wedge y = m)) \\
& \quad \quad \quad \rightarrow \chi_{P,3}(f, x, y, p) \wedge \\
& \quad \quad \forall x \land \bigwedge_{i=1}^6 \chi_{T,3}(f, x, p)
\end{align*}
\]

In order to understand \( \Phi_A \) above, we first note that

\[
\psi(f, p) \rightarrow \chi_{P,1}(f, c, p) \rightarrow \neg \psi(f, p)
\]

Now suppose \( \Phi_A(f_1, p) \) and \( \Phi_A(f_2, p) \) are true for \( f_1 \neq f_2 \). Further suppose \( \psi(f_1, p) \) and \( \psi(f_2, p) \) are true.

Consider \( f_1 \). Since \( \Phi_A(f_1, p) \) is true we have \( \exists f ((f \neq f_1) \wedge \psi(f, p)) \rightarrow \Psi(f_1, p) \). Now as \( f_2 \neq f_1 \) and \( \psi(f_2, p), \Psi(f_1, p) \) is true. This is a contradiction as \( \psi(f_1, p) \rightarrow \neg \Psi(f_1, p) \). Thus \( \psi(f_1, p) \) and \( \psi(f_2, p) \) cannot be simultaneously true. This, coupled with the fact that

\[
\Phi_A(f, p) \rightarrow (\neg \exists f_1 ((f_1 \neq f) \wedge \psi(f_1, p)) \rightarrow \psi(f, p))
\]

ensures that if \( \Phi_A(f, p) \) is true for some \( f \), then there is exactly one \( f^* \) such that \( \Phi_A(f^*, p) \) and \( \psi(f^*, p) \) are true. Then from \( T_{ss'}(p) \), we have

\[
\begin{align*}
\neg \Phi_G(f, p) & \rightarrow \Psi(f, p) \rightarrow \neg \psi(f, p) \\
\Phi_G(f, p) & \rightarrow \Phi_A(f, p) \rightarrow \exists ! f^* \psi(f^*, p)
\end{align*}
\]
Thus if there is some \( f_{\xi_1} \) for which \( \Phi_G^{\xi_1}(f_{\xi_1}, p) \) is true, then there is a unique \( f_{\xi_1}^* \) such that \( \Phi_G^{\xi_1}(f_{\xi_1}^*, p) \) and \( \psi_{\xi_1}(f_{\xi_1}^*, p) \) are true. This \( f_{\xi_1}^* \) gives the particular \( b, s \) and \( c \) nodes that are chosen for the graph transformation as described above in (1) and \( \psi_{\xi_1}(f_{\xi_1}^*, p) \) gives the sub-actions to be taken on all nodes and edges.

We can similarly write the GA pair \( \xi_2 \) for (2) above.

\[
\xi_2 = (\Phi_G, \Phi_A), f = (c).
\]

\[
\Phi_G(f, p) = \mathcal{M}_s(m) \land \mathcal{T}_a(t) \land \mathcal{C}_s(c) \land N_s(m) \land N_s(t) \land \neg N_s(c)
\]

\[
\Phi_A(f, p) = \exists f_2 ((f_2 \neq f) \land \psi(f_2, p)) \rightarrow \Psi(f, p) \land
\]

\[
\psi(f, p) = \forall x, y ((x = c \land y = t) \lor \neg (x = c \land y = m))
\]

Once again as reasoned earlier, if there is some \( f_{\xi_2} \) for which \( \Phi_G^{\xi_2}(f_{\xi_2}, p) \) is true, there is a unique \( f_{\xi_2}^* \) such that \( \Phi_G^{\xi_2}(f_{\xi_2}^*, p) \) and \( \psi_{\xi_2}(f_{\xi_2}^*, p) \) are true. This \( f_{\xi_2}^* \) gives the particular \( c \) node that is chosen for the graph transformation as described above in (2) and \( \psi_{\xi_2}(f_{\xi_2}^*, p) \) gives the sub-actions to be taken on all nodes and edges.

We can now write the other operations in a similar manner. In all of them, \( \Phi_A^\xi(f, p) \) is of the form

\[
\exists f ((f \neq f) \land \psi(f, p)) \rightarrow \Psi(f, p) \land
\]

\[
\neg \exists f ((f \neq f) \land \psi(f, p)) \rightarrow \psi(f, p)
\]

Hence in the following only \( \Phi_G^\xi \) and \( \psi_\xi \) have been mentioned. For better readability, we shall omit the ’s’ subscript for e.g. instead of \( \mathcal{M}_s \) we shall simply write \( \mathcal{M} \) and similarly for others. Also instead of \( \chi_{P,i} \) we shall simply write \( \chi_i \) and for \( \gamma_{P,i} \) we shall write \( \gamma_i \).

**Borrow**\((b, m)\):

Construct \( O_{\text{Borrow}} = \{\{\xi_1, \xi_2\}, p\} \) where \( p = (b, m) \) as follows:
• $\xi_1 = (\Phi_G, \Phi_A), f = (s, l)$.

$$\Phi_G(f, p) = B(b) \land M(m) \land S(s) \land L(l)$$

$$\land N(b) \land N(m) \land N(s) \land \lnot N(l) \land E(b, s)$$

$$\psi(f, p) = \chi_1(f, l, p) \land$$

$$\land \chi_2(f, c, p) \land$$

$$\land \forall x ((x \neq l) \rightarrow \chi_3(f, x, p)) \land$$

$$\land \forall x, y ((x = b \land y = s) \lor (x = b \land y = l) \lor (x = l \land y = m))$$

$$\lor \Rightarrow \land \forall x \bigwedge_{i=1}^{i=6} \chi_{T_i, A}(f, x, p)$$

• $\xi_2 = (\Phi_G, \Phi_A), f = (c, l)$.

$$\Phi_G(f, p) = B(b) \land M(m) \land C(c) \land L(l)$$

$$\land N(b) \land N(m) \land N(c) \land \lnot N(l) \land E(b, c) \land E(c, m)$$

$$\chi_1(f, l, p) \land \chi_2(f, c, p) \land$$

$$\land \forall x ((x \neq l) \land x \neq c) \rightarrow \chi_3(f, x, p)) \land$$

$$\land \forall x, y ((x = b \land y = c) \lor (x = c \land y = m) \lor$$

$$(x = b \land y = l) \lor (x = l \land y = m)) \rightarrow \Rightarrow \land \forall x \bigwedge_{i=1}^{i=6} \chi_{T_i, A}(f, x, p)$$

Return(b, m):

Construct $O_{\text{return}} = \{\xi_1, \xi_2, \xi_3\}, p$ where $p = (b, m)$ as follows:

• $\xi_1 = (\Phi_G, \Phi_A), f = (s, l)$.

$$\Phi_G(f, p) = B(b) \land M(m) \land S(s) \land L(l)$$

$$\land N(b) \land N(m) \land N(s) \land \lnot N(l) \land E(b, s)$$

$$\land \lnot \exists \xi, \xi T(c) \land T(t) \land N(c) \land N(t) \land E(c, t) \land E(b, t))$$

$$\chi_2(f, l, p) \land$$

$$\land \forall x ((x \neq l) \rightarrow \chi_3(f, x, p)) \land$$

$$\land \forall x, y ((x = b \land y = s) \lor (x = b \land y = l) \lor (x = l \land y = m))$$

$$\lor \Rightarrow \land \forall x \bigwedge_{i=1}^{i=6} \chi_{T_i, A}(f, x, p)$$
Construct $O_{\text{Addbook}} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$ where $p = (b, t, s)$ as follows:

- $\xi_2 = (\Phi_G, \Phi_A), f = (s, l)$.

\[
\Phi_G(f, p) = \begin{array}{c}
B(b) \land M(m) \land S(s) \land L(l) \\
N(b) \land N(m) \land N(l) \land \neg N(s) \\
\neg \exists s_1(S(s_1) \land N(s_1)) \\
E(b, l) \land E(l, m) \land \neg E(b, s) \\
\neg \exists c, t(C(c) \land T(t) \land N(c) \land N(t) \land E(c, t) \land E(b, t)) \\
\chi_2(f, l, p) \land \chi_1(f, s, p) \\
\Upsilon_2(f, b, l, p) \land \Upsilon_2(f, l, m, p) \land \Upsilon_2(f, b, s, p) \land \\
\forall x ((x \neq l) \rightarrow \chi_3(f, x, p)) \land \\
\forall x, y - ((x = b \land y = s) \lor (x = b \land y = l) \lor (x = l \land y = m)) \\
\rightarrow \Upsilon_3(f, x, y, p) \land \\
\forall x \land \bigwedge_{i=1}^{i=6} \chi_{T, 3}(f, x, p)
\end{array}
\]

- $\xi_3 = (\Phi_G, \Phi_A), f = (c, l, t)$.

\[
\Phi_G(f, p) = \begin{array}{c}
B(b) \land M(m) \land C(c) \land L(l) \land T(t) \\
N(b) \land N(m) \land N(c) \land N(l) \land \neg N(t) \\
E(b, t) \land E(c, t) \land E(b, l) \land E(l, m) \\
\chi_2(f, l, p) \land \\
\Upsilon_1(f, b, c, p) \land \\
\Upsilon_2(f, b, l, p) \land \Upsilon_2(f, l, m, p) \land \Upsilon_2(f, c, t, p) \land \\
\forall x ((x \neq l) \rightarrow \chi_3(f, x, p)) \land \\
\forall x, y - ((x = b \land y = c) \lor (x = c \land y = c)) \lor (x = b \land y = t) \lor (x = l \land y = m) \lor (x = c \land y = m) \\
\rightarrow \Upsilon_3(f, x, y, p) \land \\
\forall x \land \bigwedge_{i=1}^{i=6} \chi_{T, 3}(f, x, p)
\end{array}
\]

\[
\text{Addbook}(b, t, s): \\
\forall x ((x \neq b) \rightarrow \chi_3(f, x, p)) \land \\
\forall x, y - ((x = b \land y = t) \lor (x = b \land y = s)) \rightarrow \Upsilon_3(f, x, y, p) \land \\
\forall x \land \bigwedge_{i=1}^{i=6} \chi_{T, 3}(f, x, p)
\]
\[\xi_2 = (\Phi_G, \Phi_A), \mathbf{f} = (d).\]

\[\Phi_G(\mathbf{f}, \mathbf{p}) = B(b) \land T(t) \land S(s) \land \neg N(b) \land \neg N(t) \land N(s) \]
\[\quad \land \neg \exists c(C(c) \land N(c) \land E(c, t)) \]

\[
\psi(\mathbf{f}, \mathbf{p}) = \chi_1(f, b, p) \land \chi_1(f, t, p) \land \chi_1(f, s, p) \land \\
\quad \land \Upsilon_1(f, b, t, p) \land \Upsilon_1(f, b, s, p) \land \\
\quad \land \forall x ((x = b \land x = t) \rightarrow \chi_3(f, x, p)) \land \\
\quad \land \forall x, y (\neg ((x = b \land y = t) \lor (x = b \land y = s)) \rightarrow \Upsilon_3(f, x, y, p) \land \\
\quad \land \forall x \land_{i=1}^{i=6} \chi_{T, 3}(f, x, p) \]

\[\xi_3 = (\Phi_G, \Phi_A), \mathbf{f} = (d).\]

\[\Phi_G(\mathbf{f}, \mathbf{p}) = B(b) \land T(t) \land S(s) \land \neg N(b) \land N(t) \land \neg N(s) \land \neg N(t) \land N(s) \land \neg \exists c(C(c) \land N(c) \land E(c, t)) \]

\[
\psi(\mathbf{f}, \mathbf{p}) = \chi_1(f, b, p) \land \chi_1(f, s, p) \land \\
\quad \land \Upsilon_1(f, b, t, p) \land \Upsilon_1(f, b, s, p) \land \\
\quad \land \forall x ((x \neq b \land x \neq s) \rightarrow \chi_3(f, x, p)) \land \\
\quad \land \forall x, y (\neg ((x = b \land y = t) \lor (x = b \land y = s)) \rightarrow \Upsilon_3(f, x, y, p) \land \\
\quad \land \forall x \land_{i=1}^{i=6} \chi_{T, 3}(f, x, p) \]

\[\xi_4 = (\Phi_G, \Phi_A), \mathbf{f} = (d).\]

\[\Phi_G(\mathbf{f}, \mathbf{p}) = B(b) \land T(t) \land S(s) \land \neg N(b) \land \neg N(t) \land \neg N(s) \land \neg N(t) \land N(s) \land \neg \exists c(C(c) \land N(c) \land E(c, t)) \]

\[
\psi(\mathbf{f}, \mathbf{p}) = \chi_1(f, b, p) \land \chi_1(f, t, p) \land \chi_1(f, s, p) \land \\
\quad \land \Upsilon_1(f, b, t, p) \land \Upsilon_1(f, b, s, p) \land \\
\quad \land \forall x ((x \neq b \land x \neq s \land x \neq t) \rightarrow \chi_3(f, x, p)) \land \\
\quad \land \forall x, y (\neg ((x = b \land y = t) \lor (x = b \land y = s)) \rightarrow \Upsilon_3(f, x, y, p) \land \\
\quad \land \forall x \land_{i=1}^{i=6} \chi_{N, T, 3}(f, x, p) \]
\[ \xi_5 = (\Phi_G, \Phi_A), f = (c). \]

\[ \Phi_G(f, p) = B(b) \wedge T(t) \wedge \neg N(b) \wedge N(t) \]
\[ C(c) \wedge N(c) \wedge E(c, t) \]

\[ \psi(f, p) = \chi_1(f, b, p) \wedge \]
\[ \forall x \left( (x \neq b) \rightarrow \chi_3(f, x, p) \right) \wedge \]
\[ \forall x, y \left( (x = b \wedge y = t) \vee (x = b \wedge y = c) \vee (x = c \wedge y = t) \right) \]
\[ \rightarrow \chi_3(f, x, y, p) \wedge \]
\[ \forall x \wedge_{i=1}^{i=6} \chi_{N, i, 3}(f, x, p) \]

**AddMember(m):**

Construct \( O_{Addmember} = (\{\xi_1\}, p) \) where \( p = (m) \) as follows:

\[ \xi_1 = (\Phi_G, \Phi_A), f = (d). \]

\[ \Phi_G(f, p) = M(m) \wedge \neg N(m) \]

\[ \psi(f, p) = \chi_1(f, m, p) \wedge \]
\[ \forall x \left( (x \neq m) \rightarrow \chi_3(f, x, p) \right) \wedge \]
\[ \forall x, y \left( \chi_{N, x, y}(f, x, y, p) \wedge \right. \]
\[ \forall x \wedge_{i=1}^{i=6} \chi_{N, i, 3}(f, x, p) \]

Having written the operations, we can write the GTS, \( S_{Library} \) for the library system as

\[ S_{Library} = (\Delta_{Library}, \rho), \]

where

\[ \Delta_{Library} = \{O_{Addbook}, O_{Addmember}, O_{Borrow}, O_{Return}, O_{Reserve}\} \]

We now make the following observations about \( S_{Library} \). For each operation \( O = (\Omega, p) \in \Delta_{Library} \), for each \( \xi \in \Omega \) we have the following:

1. \( \Phi_A(f, p) \) is of the form

   \[ \exists f_1 ((f_1 \neq f) \wedge \psi(f_1, p)) \rightarrow \psi(f, p) \wedge \]
   \[ \neg \exists f_1 ((f_1 \neq f) \wedge \psi(f_1, p)) \rightarrow \psi(f, p) \]

2. \( \psi(f, p) \rightarrow \neg \psi(f, p) \)

3. \( \psi(f, p) \) is a \( \forall^* \) formula in prenex normal form. By Claim 3 then, given any \( \sigma_{A_1}^O \)

structure \( M \) with universe \( A \) such that \( M \models \psi(f_1, p) \) where \( f_1 \in A[\rho], p_1 \in A[p] \),

there exists \( A_1 \subseteq A \) such that
9.1. CLASS 1

(a) \( f_1 \in A_1^{P}, p_1 \in A_1^{P} \)

(b) \( |A_1| \leq ||f_1|| + |p_1| \) (hence \( A_1 \) is of bounded size)

(c) \( \forall A_2, A_1 \subseteq A_2 \subseteq A \), the substructure \( M_2 \) of \( M \) generated by \( A_2 \) is such that \( M_2 \models \psi(f_1, p_1) \).

4. \( \Phi_G(f, p) \) has the following properties:

(a) It is a \( \forall^* \) formula in prenex normal form.

(b) No occurrence of each of \( x = y \) and \( E_s(x, y) \) is such that one of the variables is quantified and the other is in \( f \). Thus both \( x \) and \( y \) are such that either (a) each is quantified or is in \( p \) or (b) each is free (i.e. belongs to \( f \) or \( p \)).

We later show that for \( \Phi_G \) of the form stated, given a \( \sigma_s \)-structure \( M \) with universe \( A \) and any \( p_1 \in A^{P} \), there exists \( A_1 \subseteq A \) such that

(a) \( p_1 \in A_1^{P} \)

(b) \( A_1 \) is of bounded size

(c) \( \forall A_2, A_1 \subseteq A_2 \subseteq A \), the substructure \( M_2 \) of \( M \) generated by \( A_2 \) is such that

\[
M_2 \models \Phi_G(f_1, p_1) \iff M \models \Phi_G(f_1, p_1), f_1 \in A_2^{P}.
\]

In particular then, if \( M \models \forall f \neg \Phi_G(f, p_1) \), then

\[
M_2 \models \forall f \neg \Phi_G(f, p_1).
\]

We now prove the following result for a generalisation of the particular example of the Library system considered above.

We will require the following definition. Let \( \text{Sig}(\varphi) \) be the set of all predicate and constant symbols occurring in a FOL formula (with equality) \( \varphi \).

**Definition 9.1 (EBS property)** A formula \( \varphi(x) \) is said to satisfy the Extensible Bounded Submodel property with respect to \( \sigma \subseteq \Sigma \) (denoted as \( \varphi(x) \in \text{EBS}_\Sigma(\sigma) \)) if \( \text{Sig}(\varphi) = \Sigma \) and there exists a cardinal \( \mathcal{B} \) (dependent on \( \varphi \) and \( \sigma \) in general) such that for every \( \Sigma \)-structure \( M \) having universe \( A \) with \( |A| \geq \mathcal{B} \) and for every \( a \in A^{|\Sigma|} \), if \( M \models \varphi(a) \) then there exists \( A_1 \subseteq A \) such that

1. \( a \in A_1^{|\Sigma|} \)

2. \( |A_1| \leq \mathcal{B} \)

3. For all \( A_2 \) such that \( A_1 \subseteq A_2 \subseteq A \), if \( M_2' \) is the substructure of \( M \) generated by \( A_2 \), then there exists a \( \Sigma \)-structure \( M_2 \) such that \( M_2|_{\sigma} = M_2'|_{\sigma} \) and \( M_2 \models \varphi(a) \).

Note that in the special case that \( \Sigma = \sigma \), the condition (3) above reduces to

\[
\forall A_2 \text{ such that } A_1 \subseteq A_2 \subseteq A, \text{ if } M_2 \text{ is the substructure of } M \text{ generated by } A_2, \text{ then } M_2 \models \psi(a).
\]
9.1.2 Generalization

Result 1:

Consider a GTS $S = (\Delta, \rho)$ in which for every operation $O = (\Omega, p) \in \Delta$, each GA pair $\xi \in \Omega$ satisfies the following properties:

1. $\Phi_A(f, p)$ is of the form
   \[ \exists f_1 \left((f_1 \neq f) \land \psi(f_1, p)\right) \rightarrow \Psi(f, p) \land \neg \exists f_1 \left((f_1 \neq f) \land \psi(f_1, p)\right) \rightarrow \psi(f, p) \]
   where $\psi(f, p)$ is a formula over $\sigma^O$ satisfying the following
      (a) $\psi(f, p) \rightarrow \neg \psi(f, p)$
      (b) $\psi(f, p) \in \text{EBS}_\Sigma(\Sigma)$ where $\Sigma = \sigma^O$.

2. $\Phi_G(f, p)$ is such that
   (a) $\Phi_G(f, p) \in \text{EBS}_\Sigma(\Sigma)$ where $\Sigma = \sigma_s$.
   (b) $\forall f \neg \Phi_G(f, p) \in \text{EBS}_\Sigma(\Sigma)$ where $\Sigma = \sigma_s$.

Then $T_{ss'}^O(p) \in \text{EBS}_\Sigma(\Sigma)$ where $\Sigma = \sigma_{ss'}^O$.

Proof:

Firstly we prove the following:

Claim 4:

For each $\xi \in \Omega$, either

1. $M \models \neg \Phi_G(f_1, p_1)$ (and hence $M \models \Psi(f_1, p_1)$) $\forall f_1 \in \text{A}^{[f]}$ or
2. There is a unique $f_1^*$ such that $M \models (\Phi_G(f_1^*, p_1) \land \psi(f_1^*, p_1))$ and $\forall f_1 \neq f_1^*, M \models \Psi(f_1, p_1)$

Proof:

Let $S = \{f_1 \in \text{A}^{[f]} | M \models \Phi_G(f_1, p_1)\}$.

For any $f_1 \in \text{A}^{[f]}$, we have

\[ M \models \neg \Phi_G(f_1, p_1) \]
\[ \models \Psi(f_1, p_1) \quad (\because M \models T_{ss'}^O(p_1)) \]
\[ \models \neg \psi(f_1, p_1) \quad (\text{by 1(a) in the statement of the Result}) \]
1. If $S = \{\}$, then for each $f_1 \in A^{M}$, $M \models \neg \Phi_G(f_1, p_1)$ and thus from above, $M \models \Psi(f_1, p_1)$.

2. Else suppose that $M \models \psi(f_1, p_1), \forall f_1 \in S$. (For $f_1 \notin S$, $M \models \neg \psi(f_1, p_1)$ as shown above.) Then by 1(b) in the statement of the result, $M \models T_{ss}^O(p_1)$.

Similarly for each $S;M$ giving a contradiction. Hence there exists $f_1 \in S$, $M \models \Phi_G(f_1, p_1)$

$$M \models \Phi_G(f_1, p_1)$$

$$\models \Phi_A(f_1, p_1)$$

$$\models \neg \exists f((f \neq f_1) \land \psi(f, p_1)) \rightarrow \psi(f_1, p_1)$$

$$\models \psi(f_1, p_1)$$

giving a contradiction.

Thus there exists a unique $f^*_\xi \in S$ such that $M \models \psi(f^*_\xi, p_1)$ and for $f_1 \neq f^*_\xi, f_1 \in S, M \models \psi(f_1, p_1)$. For $f_1 \notin S, M \models \psi(f_1, p_1)$ as already shown above. Thus $M \models \psi(f_1, p_1), \forall f_1 \neq f^*_\xi$.

$$\Box$$

Let

$$S = \{ \xi | M \models \exists \Psi_G^\xi(f_1, p_1) \}$$

From Claim 4, for each $\xi \in S$, there is a unique $f^*_\xi$ such that $M \models \Phi_G(f^*_\xi, p_1) \land \psi(f^*_\xi, p_1)$.

Since $\psi$ is a $\sigma_\xi^O$ formula, from Claim 1 we have, for each $\xi \in S$,

$$M|_{\sigma_\xi^O} \models \psi(f^*_\xi, p_1)$$

Then by 1(b) in the statement of the result, $\psi(f^*_\xi, p) \in \text{EBS}_\Sigma(\Sigma)$ where $\Sigma = \sigma_\xi^O$. Then there exists $B_{\xi,1}$ such that since $M \models \psi(f^*_\xi, p_1)$ there exists $A_{\xi,1} \subseteq A(f^*_\xi \in A_{\xi,1}^{f_1}, p_1 \in A_{\xi,1}^{p_1})$ satisfying the conditions of the property. Similarly for each $\xi \in S$, by 2(a) of the statement of the result, $\Phi_G(f^*_\xi, p) \in \text{EBS}_\Sigma(\Sigma)$.
where $\Sigma = \sigma_s$. Then there exists $B_{\xi,2}$ such that since $M \models \Phi_G(f^\xi_1, p_1)$ there exists $A_{\xi,2} \subseteq A(f^\xi_1 \in A_{\xi,2}^{|\xi|}, p_1 \in A_{\xi,2}^{|p|})$ satisfying the conditions of the property. Similarly for each $\xi \notin S$, by 2(b) of the statement of the result, $\forall f - \Phi_G(f, p) \in EBS_{\xi}(\Sigma)$ where $\Sigma = \sigma_s$. Then there exists $B_{\xi}$ such that since $M \models \forall f - \Phi_G(f, p_1)$ there exists $A_{\xi} \subseteq A(p_1 \in A_{\xi}^{|p|})$ satisfying the conditions of the property.

Then construct $A_1 = \bigcup_{\xi \in S} A_{\xi,1} \cup \bigcup_{\xi \in S} A_{\xi,2} \cup \bigcup_{\xi \notin S} A_{\xi}$

Let $A_2$ be such that $A_1 \subseteq A_2 \subseteq A$. Let $M_2$ be the substructure of $M$ generated by $A_2$. Note that $p_1 \in A_1^{|p|}$.

Consider $\xi \in S$.

From above,

$M|_{\sigma^\Omega_1} \models \psi(f^\xi_1, p_1)$

Consider $M_2|_{\sigma^\Omega_1}$ which is the substructure of $M|_{\sigma^\Omega_1}$ generated by $A_2$. Since $A_{\xi,1} \subseteq A_2$, by the EBS property for $\psi(f_1, p)$,

$M_2|_{\sigma^\Omega_1} \models \psi(f^\xi_1, p_1)$

hence by Claim 1

$M_2 \models \psi(f^\xi_1, p_1) \quad \xi \in S \quad (9.1)$

Similarly,

$M_2 \models \Phi_G(f^\xi_1, p_1) \quad \xi \in S \quad (9.2)$

$M_2 \models \forall f - \Phi_G(f, p_1) \quad \xi \notin S \quad (9.3)$

For all $\xi \in \Omega, \Psi_\xi$ is a $\forall^*$ formula. Since $p_1 \in A_1^{|p|}$ and $A_1 \subseteq A_2$, $p_1 \in A_2^{|p|}$. Then from Claim 3, since $M_2$ is a substructure of $M$,

$M \models \Psi(f_1, p_1) \rightarrow M_2 \models \Psi(f_1, p_1) \quad f_1 \in A_2^{|f|} \quad (9.4)$

Then from Claim 4, 9.1, 9.4, we get

$M_2 \models \left\{ \begin{array}{ll}
\Psi(f_1, p_1) & f_1 \in A_2^{|f|}, f_1 \neq f^\xi_1, \xi \in S \\
\psi(f_1, p_1) & f_1 = f^\xi_1, f_1 \notin S \\
\Psi(f_1, p_1) & f_1 \in A_2^{|f|}, \xi \notin S
\end{array} \right. \quad (9.5)$

Consider an arbitrary $\xi \in \Omega$. We show that $M_2 \models T^\xi_{ss}(p_1)$. 


9.1. CLASS 1

Then, for $\Psi(.) \rightarrow \neg \psi(.)$, from 9.5 above, we have

$$M_2 \models \begin{cases} \exists f (f \neq f_1 \land \psi(f, p_1)) \land \Psi(f_1, p_1) & f_1 \in A^{|f|}, f_1 \neq f_\xi \\ \neg \exists f (f \neq f_1 \land \psi(f, p_1)) \land \Psi(f_1, p_1) & f_1 = f_\xi \end{cases}$$

Thus $M_2 \models \Phi_A(f, p_1) \forall f_1 \in A^{|f|}$.

Now,

If $M_2 \models \Phi_G(f_1, p_1)$, then from above $M_2 \models \Phi_A(f_1, p_1)$.

If $M_2 \models \neg \Phi_G(f_1, p_1)$, then from 9.2, $f_1 \neq f_\xi$ and hence from 9.5, $M_2 \models \Psi(f_1, p_1)$.

Thus

$$M_2 \models T^\xi_{ss'}(p_1)$$

- **Case 1:** $\xi \in S$

  Since $\Psi(.) \rightarrow \neg \psi(.)$, from 9.5 above, we have

  $$M_2 \models \begin{cases} \exists f (f \neq f_1 \land \psi(f, p_1)) \land \Psi(f_1, p_1) & f_1 \in A^{|f|}, f_1 \neq f_\xi \\ \neg \exists f (f \neq f_1 \land \psi(f, p_1)) \land \Psi(f_1, p_1) & f_1 = f_\xi \end{cases}$$

  Then from 9.3, $M_2 \models \forall f \neg \Phi_G(f, p_1)$

  and from 9.5, $M_2 \models \forall f \Psi(f, p_1)$

  and hence

  $$M_2 \models T^\xi_{ss'}(p_1)$$

  Thus from both the above cases we have

  $$M_2 \models \bigwedge_{\xi \in \Omega} T^\xi_{ss'}(p_1) \quad (9.6)$$

  Let,

  $$U_n(x, p) = \bigvee_{\xi \in \Omega} \exists f \chi^\xi_n(f, x, p) \quad \lambda_n \in \rho_N$$

  $$V_e(x, y, p) = \bigvee_{\xi \in \Omega} \exists f \gamma^\xi_e(f, x, y, p) \quad \mu_e \in \rho_E$$

  We recall that $\Psi(f, p) = \left( \forall x \left( \text{\lambda}_n \in \rho_N \rightarrow \chi^\xi_n(f, x, p) \right) \land \left( \forall x, y \left( \text{\lambda}_e \in \rho_E \rightarrow \gamma^\xi_e(f, x, y, p) \right) \right) \right)$.

  Then from Claim 4, for $\lambda_n \in \rho_N, \mu_e \in \rho_E, x, y \in A$, we have

  $$M \models \begin{cases} \neg \chi^\xi_n(f_1, x, p_1) & f_1 \in A^{|f|}, \xi \notin S \\ \neg \gamma^\xi_e(f_1, x, y, p_1) & f_1 \in A^{|f|}, \xi \notin S \\ \neg \chi^\xi_n(f_1, x, p_1) & f_1 \in A^{|f|}, f_1 \neq f_\xi \land \xi \notin S \\ \neg \gamma^\xi_e(f_1, x, y, p_1) & f_1 \in A^{|f|}, f_1 \neq f_\xi \land \xi \in S \end{cases}$$

  Then, for $x, y \in A$, the following holds in $M$.

  $$U_n(x, p_1) = \bigvee_{\xi \in S} \chi^\xi_n(f_\xi^* x, p_1) \quad \lambda_n \in \rho_N$$

  $$V_e(x, y, p_1) = \bigvee_{\xi \in S} \gamma^\xi_e(f_\xi^* x, y, p_1) \quad \mu_e \in \rho_E$$
We can similarly show using 9.5 that the above relation holds in $M_2$ as well. Thus in $M$ and $M_2$, $U_n(x, p_1)$ and $V_e(x, y, p_1)$ are in fact quantifier-free formulae.

Consider the formulae $\chi_n(x, p), \alpha_n(x, p)(\lambda_n \in \rho_N)$ and $\Upsilon_e(x, y, p), \beta_e(x, y, p)(\mu_e \in \rho_E)$. These are quantifier free formulae. Since $M_2 \subseteq M$ by Claim 2 and Claim 1, for $x, y \in A_2, \lambda_n \in \rho_N, \mu_e \in \rho_E$

\[
\begin{align*}
M_2 \models \chi_n(x, p_1) & \iff M \models \chi_n(x, p_1) \\
M_2 \models \Upsilon_e(x, y, p_1) & \iff M \models \Upsilon_e(x, y, p_1) \\
M_2 \models U_n(x, p_1) & \iff M \models U_n(x, p_1) \\
M_2 \models V_e(x, y, p_1) & \iff M \models V_e(x, y, p_1) \\
M_2 \models \alpha_n(x, p_1) & \iff M \models \alpha_n(x, p_1) \\
M_2 \models \beta_e(x, y, p_1) & \iff M \models \beta_e(x, y, p_1)
\end{align*}
\]

Using the above and the fact that since $M \models T_{ss}^O(p_1), M \models Z^O(p_1) \land \Gamma^O(p_1)$, we have

\[M_2 \models Z^O(p_1) \land \Gamma^O(p_1) \quad (9.7)\]

From 9.5 and 9.6,

\[M_2 \models T_{ss}^O(p_1)\]

Note that $A_1 \subseteq A_2$ and $p_1 \in A^{[p]}$

\[\square\]

Bound on $|A_1|$: 

\[
A_1 = \bigcup_{\xi \in S}(A_{\xi,1} \cup A_{\xi,2}) \cup \bigcup_{\xi \notin S} A_{\xi}
\]

\[
|A_1| \leq \Sigma_{\xi \in S}(|A_{\xi,1}| + |A_{\xi,2}|) + \Sigma_{\xi \notin S}|A_{\xi}|
\]

\[
\leq \Sigma_{\xi \in S}(B_{\xi,1} + B_{\xi,2}) + \Sigma_{\xi \notin S}B_{\xi}
\]

\[
\leq \max_{\xi \subseteq \Omega} \{\Sigma_{\xi \in S}(B_{\xi,1} + B_{\xi,2}) + \Sigma_{\xi \notin S}B_{\xi}\}
\]

\[= B_O(say)\]

**Corollary 1:**

Consider a GTS $S = (\Delta, \rho)$ where each operation has the properties stated in the previous result. Then $T_{ss'} \in EBS_\Sigma(\Sigma)$ where $\Sigma = \sigma_{ss'}$. 
Proof:

\[ T_{ss'} = \bigvee_{O=(\Omega,p) \in \Delta} \exists p T_{ss'}^O(p) \]

Since \( M \models T_{ss'} \), \( M \models T_{ss'}^O(p_1) \) for some \( O \in \Delta \) and some \( p_1 \in A^{[p]} \). By Claim 1, \( M|_{\sigma_{ss'}} \models T_{ss'}^O(p_1) \). Then by Result 1, since \( T_{ss'}^O(p) \in \text{EBS}_{\Sigma}(\Sigma) \) where \( \Sigma = \sigma_{ss'}^O \), there exists a \( B_O \) such that since \( M|_{\sigma_{ss'}} \models T_{ss'}^O(p_1) \) there exists \( A_1 \subseteq A \) such that

1. \( |A_1| \leq B_O \)
2. \( p_1 \in A_1^{[p]} \)
3. \( \forall A_2, A_1 \subseteq A_2 \subseteq A \), the substructure \( M_2' \) of \( M|_{\sigma_{ss'}} \) generated by \( A_2 \) is such that \( M_2' \models T_{ss'}^O(p_1) \).

Then consider the substructure \( M_2 \) of \( M \) generated by \( A_2 \). Clearly \( M_2|_{\sigma_{ss'}} = M_2' \). Since \( M_2' \models T_{ss'}^O(p_1) \) by Claim 1, we have \( M_2 \models T_{ss'}^O(p_1) \) and hence \( M_2 \models T_{ss'} \).

\[ |A_1| \leq B_O \leq \max_{O \in \Delta} B_O \ (= B) \]

\[ \square \]

9.2 Class 2

9.2.1 An Example

Before considering the example, we need the following definition:

Let \( \sigma \) be a vocabulary. Let \( \varphi(f) \) be a formula over \( \sigma \). Then \( \varphi(f) \) is said to be positive unate in a \( k \)-ary predicate \( P \in \sigma \) if the following is true:

Let \( M \) be any \( \sigma \)-structure (with universe \( A \)) s.t. \( M \models \varphi(f_1), (f_1 \in A^{[f]} \). Then each \( \sigma \)-structure \( M_1 \) with the same universe \( A \) and which differs from \( M \) only in the value of \( P(x_1, x_2, \ldots, x_k) \) being false in \( M \) and true in \( M_1 \), is such that \( M_1 \models \varphi(f_1) \).

Now consider a GTS \( S = (\Delta, \rho) \), where for each operation \( O = (\Omega, p) \in \Delta \), each GA pair \( \xi \in \Omega \), has the following properties:

1. \( \Phi_G(f, p) \) has the following properties:

   (a) It is a \( \exists^* \) (or a \( \forall^* \)) formula in prenex normal form.
(b) No occurrence of each of \( x = y, E_s(x, y) \) and \( T_{E,s}^i(x, y) (1 \leq i \leq N_e) \) is such that one of the variables is quantified and the other is in \( f \). Thus both \( x \) and \( y \) are such that either (i) each is quantified or is in \( p \) or (ii) each is free (i.e. belongs to \( f \) or \( p \)).

2. \( \Phi_A(f, p) \) is a \( \forall^* \) formula in prenex normal form and is positive unate in the predicates of \( \sigma_N^{\alpha_l} \cup \sigma_E^{\alpha_l} \) (i.e. \( \{ x_\xi | x_n \in \rho_N, \xi \in \Omega \} \cup \{ T_\xi_{\mu_e} | \mu_e \in \rho_E, \xi \in \Omega \})

We now prove the following:

\( \Phi_G \) is such that for any \( \sigma_s \)-structure \( M \) with universe \( A \) and for any \( p_1 \in A^{|p|} \), there exists a subset \( A_1 \subseteq A \) such that

1. \( A_1 \) is of bounded size
2. \( p_1 \in A_1^{|p|} \)
3. \( \forall A_2, A_1 \subseteq A_2 \subseteq A \), the substructure \( M_2 \) of \( M \) generated by \( A_2 \) is such that

\[
M_2 \models \Phi_G(f_1, p_1) \iff M \models \Phi_G(f_1, p_1) \quad \forall f_1 \in A_2^{|f|}
\]

**Proof:**

Suppose \( \Phi_G(f, p) \) is an \( \exists^* \) formula. (The case when it is a \( \forall^* \) formula is similar)

Then in disjunctive normal form, \( \Phi_G(f, p) \) can be written as

\[
\exists y \bigvee_{i=1}^{i=n} D_i(f, y, p)
\]

where \( D_1, D_2, \ldots, D_n \) are the disjuncts of the matrix of \( \Phi_G \). Each disjunct is a conjunction of atomic formulae and can be written as

\[
D_i(f, y, p) = C_i(f, p) \land B_i(y, p)
\]

where \( C_i \) is quantifier-free and has only the variables of \( f \) and \( p \) appearing in it and \( B_i \) is quantifier-free and has only the variables of \( y \) and \( p \) appearing in it.

(This follows from 1(b)(i) and 1(b)(ii) above.)

If \( y = (y_1, y_2, \ldots, y_m) \) then

\[
\Phi_G(f, p) = \exists y_1, y_2, \ldots, y_m \bigvee_{i=1}^{i=n} D_i(f, y_1, y_2, \ldots, y_m, p)
\]

As the \( \exists \) distributes over disjunction,

\[
\Phi_G(f, p) = \exists y_1, y_2, \ldots, y_{m-1} \bigvee_{i=1}^{i=n} \exists y_mD_i(f, y_1, y_2, \ldots, y_m, p)
\]
Continuing this way for the remaining $y_j$s,

$$
\Phi_G(f, p) = \bigvee_{i=1}^{i=n} \exists y D_i(f, y, p) \\
= \bigvee_{i=1}^{i=n} \exists y (C_i(f, p) \land B_i(y, p)) \\
= \bigvee_{i=1}^{i=n} (C_i(f, p) \land \exists y B_i(y, p))
$$

Now given a $\sigma_s$–structure $M$ with universe $A$ and any $p_1 \in A^{[p]}$, let

$$
S = \{ i | 1 \leq i \leq n, M \models \exists y B_i(y, p_1) \} 
$$

By Claim 3, for each $i \in S$, there exists $A_{i,1} \subseteq A$ such that $p_1 \in A_{i,1}^{[p]}$ and $|A_{i,1}| \leq ||p_1|| + m$ satisfying the properties given in the claim. Then construct

$$
A_1 = \bigcup_{i \in S} A_{i,1}
$$

Then $p_1 \in A_1^{[p]}$ and $|A_1| \leq \Sigma_{i \in S} |A_{i,1}| \leq n.(||p|| + m)$. Thus $A_1$ is bounded. Consider $A_2$ such that $A_1 \subseteq A_2 \subseteq A$ and let $M_2$ be the substructure of $M$ generated by $A_2$.

1. Since $C_i(1 \leq i \leq n)$ is a quantifier-free formula and $M_2$ is a substructure of $M$, for any $f_1 \in A_2^{[f]}$, we have from Claim 2,

$$
M_2 \models C_i(f_1, p_1) \iff M \models C_i(f_1, p_1)
$$

2. For $i \in S, A_{i,1} \subseteq A_2$. Then from Claim 3

$$
M_2 \models \exists y B_i(y, p_1)
$$

For $i \notin S$,

$$
M \models \neg \exists y B_i(y, p_1) \\
\models \forall y \neg B_i(y, p_1)
$$

Then again from Claim 3, since $p_1 \in A_1^{[p]}$, we have

$$
M_2 \models \forall y \neg B_i(y, p_1) \\
\models \neg \exists y B_i(y, p_1)
$$

Thus

$$
M \models \exists y B_i(y, p_1) \iff M_2 \models \exists y B_i(y, p_1) \quad 1 \leq i \leq n
$$
Then for any $f_1 \in A_2^{|f|}$, since $\Phi_G(f_1, p_1)$ is a boolean combination of $C_i(f_1, p_1)$ and $\exists y B_i(y, p_1)$ $(1 \leq i, j \leq n)$,

$$M \models \Phi_G(f_1, p_1) \iff M_2 \models \Phi_G(f_1, p_1) \quad \forall f_1 \in A_2^{|f|}$$

If $\Phi_G$ is a $\forall^*$ formula, then writing its matrix in conjunctive normal form and proceeding similarly, we can prove the above equivalence.

Suppose now that for a $\sigma_A^O$-structure $M$ with universe $A$, and for $p_1 \in A^{|p|}$,

$$M \models \bigvee_{\xi \in \Omega} \exists f (\Phi_G(f, p_1) \land \Phi_A(f, p_1))$$

Then there is $\xi^* \in \Omega$ and $f_{\xi^*} \in A_1^{|f|}$ such that

$$M \models \left( \Phi_G^{\xi^*}(f_{\xi^*}, p_1) \land \Phi_A^{\xi^*}(f_{\xi^*}, p_1) \right)$$

Construct $A_1$ as shown above for $\Phi_G^{\xi^*}$. $A_1$ is of bounded size. $f_{\xi^*} \in A_1^{|f|}, p_1 \in A_1^{|p|}$. Consider $A_2$ such that $A_1 \subseteq A_2 \subseteq A$. Let $M_2$ be the substructure of $M$ generated by $A_2$.

Since for all $\xi \in \Omega$, $\Phi_A$ is a $\forall^*$ formula, from Claim 3, we have

$$M \models \Phi_A(f_1, p_1) \rightarrow M_2 \models \Phi_A(f_1, p_1) \quad f_1 \in A_2^{|f|} \quad (9.8)$$

Let $A_n \subseteq A_2(\lambda_n \in \rho_N)$ and $A_e \subseteq A_2 \times A_2(\mu_e \in \rho_E)$. Then consider $M_2'$ which is exactly the same as $M_2$ except that $\chi_e^{\xi^*}(f_{\xi^*}, x, p_1)$ is set to True $(x \in A_n, \lambda_n \in \rho_N)$ and $\Upsilon_e^{\xi^*}(f_{\xi^*}, x, y, p_1)$ is set to True $(x, y) \in A_e, \mu_e \in \rho_E)$. For any $\xi \in \Omega$, since $\Phi_A$ is positive unate in the $\chi_e^{\xi^*}(\lambda_n \in \rho_N)$ and $\Upsilon_e^{\xi^*}(\mu_e \in \rho_E)$ predicates, we have

$$M_2 \models \Phi_A(f_1, p_1) \rightarrow M_2' \models \Phi_A(f_1, p_1) \quad f_1 \in A_2^{|f|} \quad (9.9)$$

From 9.8 and 9.9 above, for all $\xi \in \Omega$

$$M \models \Phi_A(f_1, p_1) \rightarrow M_2' \models \Phi_A(f_1, p_1) \quad f_1 \in A_2^{|f|}$$

We now prove the following result for a generalisation of the particular example considered above.
9.2.2 Generalisation

**Result 2:**

Consider a GTS $S = (\Delta, \rho)$ in which every operation $O = (\Omega, p)$ satisfies the following properties:

1. For any GA pair $\xi = (\Phi_G, \Phi_A) \in \Omega$, $\Phi_G$ is such that for any $\sigma_s$-structure $M$ with universe $A$ and for any $p_1 \in A^{[p]}$, there exists a subset $A_\xi \subseteq A$ such that
   
   (a) $A_\xi$ is of bounded size
   
   (b) $p_1 \in A_\xi^{[p]}$
   
   (c) $\forall A_2, A_\xi \subseteq A_2 \subseteq A$, the substructure $M_2$ of $M$ generated by $A_2$ is such that
   
   $$M_2 \models \Phi_G(f_1, p_1) \iff M \models \Phi_G(f_1, p_1) \quad \forall f_1 \in A_2^{[f]}$$

2. For any $\sigma_A$-structure $M$, for any $p_1 \in A^{[p]}$, if $M \models \bigvee_{\xi \in \Omega} \exists f (\Phi_G(f, p_1) \land \Phi_A(f, p_1))$, then there exists $\xi^* \in \Omega, f_{\xi^*} \in A^{[f]}$ such that $M \models \left(\Phi_G^\xi(f_{\xi^*}, p_1) \land \Phi_A^\xi(f_{\xi^*}, p_1)\right)$ and there exists $A_1 \subseteq A$ such that
   
   (a) $A_1$ is of bounded size
   
   (b) $f_{\xi^*} \in A_1^{[f]}$ and $p_1 \in A_1^{[p]}$
   
   (c) $\forall A_2, A_1 \subseteq A_2 \subseteq A$, the substructure $M'_2$ of $M$ generated by $A_2$ satisfies the following:

   For any subset $A_n \subseteq A_2(\lambda_n \in \rho_N)$ and any subset $A_e \subseteq A_2 \times A_2(\mu_e \in \rho_E)$, the $\sigma_A$-structure $M_2$ obtained from $M'_2$ by setting $\chi^\xi_n(f_{\xi^*}, x, p_1)$ to $\text{True}$ ($x \in A_n, \lambda_n \in \rho_N$) and $\chi^\xi_e(f_{\xi^*}, x, y, p_1)$ to $\text{True}$ ($(x, y) \in A_e, \mu_e \in \rho_E$) is such that

   $$M \models \Phi_A(f_1, p_1) \rightarrow M_2 \models \Phi_A(f_1, p_1) \quad \forall f_1 \in A_2^{[f]}, \xi \in \Omega$$

Then $T_{ss'}^\xi(p) \in \text{EBS}_\Sigma(\sigma)$ where $\Sigma = \sigma_{ss'}^\xi$ and $\sigma = \sigma_s \cup \sigma_{ss'}$.

**Proof:**

We are given $\sigma_{ss'}$-structure $M$ with universe $A$ such that $M \models T_{ss'}^\xi(p_1)$ for $p_1 \in A^{[p]}$. Consider $M|_{\sigma_s}$. Then for each $\xi \in \Omega$, for $\Phi_G$, there exists a subset $A_\xi \subseteq A$ such that $p_1 \in A_\xi^{[p]}, A_\xi$ is bounded in size ($|A_\xi| \leq B_\xi$) and satisfies the properties stated above.

1. If $M \models \neg \bigvee_{\xi \in \Omega} \exists f \Phi_G^\xi(f, p_1)$, then construct

   $$A_1 = \bigcup_{\xi \in \Omega} A_\xi$$
Note that $A_1 \subseteq A$, $p_1 \in A_1^{[p]}$. 

$|A_1| \leq \sum_{\xi \in \Omega} |A_\xi| \leq \sum_{\xi \in \Omega} |B_\xi|.

Consider $A_2$ such that $A_1 \subseteq A_2 \subseteq A$. Construct $M_2$ as the substructure of $M$ generated by $A_2$. Now, since $M \models \sigma_\xi p_1$ and $M \models \neg \bigwedge_{\xi \in \Omega} \exists f \Phi_G^\xi(f, p_1)$,

$$M \models \Psi(f_1, p_1) \quad f_1 \in A_1^{[f]}, \xi \in \Omega$$

(9.10)

Since $\Psi$ is a $\forall^*$ formula, $p_1 \in A_2^{[p]}$ and $M_2 \subseteq M$, by Claim 2,

$$M_2 \models \Psi(f_1, p_1) \quad f_1 \in A_2^{[f]}, \xi \in \Omega$$

(9.11)

Further since for each $\xi \in \Omega$, $A_\xi \subseteq A_2$ and $M_2|_{\sigma_\xi} \subseteq M|_{\sigma_\xi}$, from assumption,

$$M_2|_{\sigma_\xi} \models \Phi_G^\xi(f_1, p_1) \iff M|_{\sigma_\xi} \models \Phi_G^\xi(f_1, p_1) \quad \forall f_1 \in A_2^{[f]}$$

Then by Claim 1,

$$M_2 \models \Phi_G^\xi(f_1, p_1) \iff M \models \Phi_G^\xi(f_1, p_1) \quad \forall f_1 \in A_2^{[f]}$$

Since $M \models \neg \bigwedge_{\xi \in \Omega} \exists f \Phi_G^\xi(f, p_1)$,

$$M_2 \models \neg \Phi_G^\xi(f_1, p_1) \quad f_1 \in A_2^{[f]}, \xi \in \Omega$$

(9.12)

From 9.11 and 9.12, we can see that

$$M_2 \models T_{ss}^\xi(p_1) \quad \xi \in \Omega$$

Let

$$U_n(x, p) = \bigvee_{\xi \in \Omega} \exists f \chi_n^\xi(f, x, p) \quad \lambda_n \in \rho_N$$

$$V_e(x, y, p) = \bigvee_{\xi \in \Omega} \exists f \tau_e^\xi(f, x, y, p) \quad \mu_e \in \rho_E$$

Then from 9.10 and 9.11 we can also see that

$$M_2 \models U_n(x, p_1) \iff M \models U_n(x, p_1) \quad x \in A_2, \lambda_n \in \rho_N$$

$$M_2 \models V_e(x, y, p_1) \iff M \models V_e(x, y, p_1) \quad x, y \in A_2, \mu_e \in \rho_E$$

We shall use these facts later.
2. If $M \models \forall \xi \in \Omega \exists f \Phi_G^\xi(f, p_1)$, then since $M \models T^O_{ss}(p_1)$

$$M \models \bigwedge_{\xi \in \Omega} \forall f \left( \Phi_G^\xi(f, p_1) \rightarrow \Phi_A^\xi(f, p_1) \right)$$

and therefore $M \models \forall \xi \in \Omega \exists f \left( \Phi_G^\xi(f, p_1) \land \Phi_A^\xi(f, p_1) \right)$.

Then by assumption, there exists $\xi^* \in \Omega$ and $f_{\xi^*} \in A_1^{[n]}$ such that

$$M \models \left( \Phi_G^\xi(f_{\xi^*}, p_1) \land \Phi_A^\xi(f_{\xi^*}, p_1) \right)$$

and there exists $A_{\xi^*} \subseteq A$ which is bounded in size ($|A_{\xi^*}| \leq B_{\xi^*}$) satisfying the assumptions.

Then construct

$$A_1 = \bigcup_{\xi \in \Omega} A_{\xi} \cup A_{\xi^*}$$

Note that $A_1 \subseteq A$, $p_1 \in A_1^{[n]}$ and $|A_1| \leq (\Sigma_{\xi \in \Omega} |A_{\xi}|) + |A_{\xi^*}| \leq (\Sigma_{\xi \in \Omega} |B_{\xi} + B_{\xi^*}|)$.

Consider $A_2$ such that $A_1 \subseteq A_2 \subseteq A$. Construct the substructure $M'_2$ of $M$ generated by $A_2$.

Consider $x \in A_2$. For $\lambda_n \in \rho_N$, if $M'_2 \models \chi_n^x(f_1, x, p_1)$ for some $\xi \in \Omega$ and some $f_1 \in A_2^{[n]}$, then $M'_2 \models U_n(x, p_1)$. Then as $M'_2 \subseteq M$, $M \models \chi^x_n(f_1, x, p_1)$ and hence $M \models U_n(x, p_1)$. However vice-versa might not be true. Like-wise for $\mu_e \in \rho_E$. Hence let

$$A_n = \left\{ x \in A_2 \mid M'_2 \models \neg U_n(x, p_1) \quad \text{but} \quad M \models U_n(x, p_1) \right\}, \quad \lambda_n \in \rho_N$$

$$A_e = \left\{ (x, y) \in A_2 \times A_2 \mid M'_2 \models \neg V_e(x, y, p_1) \quad \text{but} \quad M \models V_e(x, y, p_1) \right\}, \quad \mu_e \in \rho_E$$

Consider the $\sigma^O_A$-structure $M_2$ which is the same as $M'_2$ except that $\chi_n^x(f_{\xi^*}, x, p_1)$ is set to True ($x \in A_n, \lambda_n \in \rho_N$) and $\gamma_{\xi^*}(f_{\xi^*}, x, y, p_1)$ is set to True ($(x, y) \in A_e, \mu_e \in \rho_E$). By the assumption about $\Phi_A$, for each $\xi \in \Omega$, we have

$$M \models \Phi_A(f_1, p_1) \rightarrow M \models \Phi_A(f_1, p_1) \quad \forall f_1 \in A_2^{[n]} \quad (9.13)$$

Now $M_2|_{\sigma_s} = M'_2|_{\sigma_s} \subseteq M|_{\sigma_s}$. Since for each $\xi \in \Omega$, $A_\xi \subseteq A_2$, by assumptions about $\Phi_G$,

$$M_2|_{\sigma_s} \models \Phi_G(f_1, p_1) \leftrightarrow M_{\sigma_s} \models \Phi_G(f_1, p_1) \quad \forall f_1 \in A_2^{[n]}$$

By Claim 1 then

$$M_2 \models \Phi_G(f_1, p_1) \leftrightarrow M \models \Phi_G(f_1, p_1) \quad \forall f_1 \in A_2^{[n]} \quad (9.14)$$
Since $\Psi$ is a $\forall^*$ formula, $p_1 \in A^{|\psi|}_2$ and $M' \subseteq M$, by Claim 2,

$$M \models \Psi(f_1, p_1) \rightarrow M' \models \Psi(f_1, p_1) \quad f_1 \in A^{|\psi|}_2 \quad (9.15)$$

Since $M_2$ differs from $M'_2$ only in the $\chi^\xi_n(f, x, p_1)$ and the $\Upsilon^\xi_n(f, x, p_1)$ values, we have for $f_1 \in A^{|\psi|}_2$,

$$M'_2 \models \Psi(f_1, p_1) \leftrightarrow M_2 \models \Psi(f_1, p_1) \quad (9.16)$$

From 9.13, 9.14, 9.15, 9.16, we have

$$M_2 \models T^{\xi}_{ss'}(p_1) \quad \xi \in \Omega$$

Now for any $x \in A_2$ and $\lambda_n \in \rho_N$

- If $x \in A_n$, then $M \models U_n(x, p_1)$ but $M'_2 \models \neg U_n(x, p_1)$. Since $M_2 \models \chi^\xi_n(f, x, p_1), M_2 \models U_n(x, p_1)$.
- If $x \notin A_n$, then $M \models U_n(x, p_1) \leftrightarrow M'_2 \models U_n(x, p_1)$. From the way $M_2$ was constructed, the $\chi^\xi_n(f, z, p_1)$ values were changed only for $\xi = \xi^*, f = f^*$ and $z \in A_n$. Since $x \notin A_n$, $M_2 \models U_n(x, p_1) \leftrightarrow M'_2 \models U_n(x, p_1)$

From either case above,

$$M_2 \models U_n(x, p_1) \leftrightarrow M \models U_n(x, p_1) \quad x \in A_2, \lambda_n \in \rho_N$$

Similarly,

$$M_2 \models V_e(x, y, p_1) \leftrightarrow M \models V_e(x, y, p_1) \quad x, y \in A_2, \mu_e \in \rho_E$$

In both cases (1) and (2) above we have shown that,

$$M_2 \models \bigwedge_{\xi \in \Omega} T^{\xi}_{ss'}(p_1) \quad (9.17)$$

$$M_2 \models U_n(x, p_1) \leftrightarrow M \models U_n(x, p_1) \quad x \in A_2, \lambda_n \in \rho_N$$

$$M_2 \models V_e(x, y, p_1) \leftrightarrow M \models V_e(x, y, p_1) \quad x, y \in A_2, \mu_e \in \rho_E$$

Consider the formulae $\chi_n(x, p), \alpha_n(x, p)(\lambda_n \in \rho_N)$ and $\Upsilon_e(x, y, p), \beta_e(x, y, p)(\mu_e \in \rho_E)$. These are quantifier free formulae. Now since

$$M_2|_{\sigma_s \cup \sigma_t \cup \sigma_N \cup \sigma_E} = M'_2|_{\sigma_s \cup \sigma_t \cup \sigma_N \cup \sigma_E} \subseteq M|_{\sigma_s \cup \sigma_t \cup \sigma_N \cup \sigma_E}$$
applying Claim 2 and subsequently Claim 1,

\[
\begin{align*}
M_2 \models \gamma(x, p_1) & \iff M \models \gamma(x, p_1) \quad x \in A_2, \lambda_n \in \rho_N \\
M_2 \models \gamma(x, y, p_1) & \iff M \models \gamma(x, y, p_1) \quad x, y \in A_2, \mu_e \in \rho_E \\
M_2 \models \alpha(x, p_1) & \iff M \models \alpha(x, p_1) \quad x \in A_2, \lambda_n \in \rho_N \\
M_2 \models \beta(x, y, p_1) & \iff M \models \beta(x, y, p_1) \quad x, y \in A_2, \mu_e \in \rho_E
\end{align*}
\]

Using the above and the fact that since \( M_2 \models T_{ss'}(p_1) \) and \( M_2 \models Z^O(p_1) \wedge \rho^O(p_1) \), we have

\[
M_2 \models Z^O(p_1) \wedge \rho^O(p_1)
\]

(9.18)

From 9.17 and 9.18,

\[
M_2 \models T_{ss'}(p_1)
\]

Note that \( M_2|_{\sigma_s \cup \sigma_{s'}} \subseteq M|_{\sigma_s \cup \sigma_{s'}} \) and \( A_1 \subseteq A_2 \subseteq A, p_1 \in A_1|^{p_1} \).

\[
|A_1| \leq (\Sigma_{\xi \in A} B_{\xi} + B_{\xi^*}) = B_O \text{ (say)}.
\]

\[\square\]

**Corollary 2:**

Consider a GTS \( S = (\Delta, \rho) \) where each operation has the properties stated in the previous result. Then \( T_{ss'} \in \text{EBS}_{\Sigma}(\sigma) \) where \( \Sigma = \sigma_{ss'} \) and \( \sigma = \sigma_s \cup \sigma_{s'} \).

**Proof:**

\[
T_{ss'} = \bigvee_{O \in (\Omega, p) \subseteq \Delta} \exists p T_{ss'}^O(p)
\]

Since \( M \models T_{ss'} \), \( M \models T_{ss'}^O(p) \) for some \( O \in \Delta \) and some \( p_1 \in A_1|^{p_1} \). By Claim 1, \( M|_{\sigma_{ss'}} \models T_{ss'}^O(p_1) \). Then by Result 2, there exists \( A_1 \subseteq A \) such that

1. \( |A_1| \leq B_O \)
2. \( p_1 \in A_1|^{p_1} \)
3. \( \forall A_2, A_1 \subseteq A_2 \subseteq A \), there exists a \( \sigma^O_{ss'} \) structure \( M_2' \) with universe \( A_2 \) such that

   (a) \( M_2'|_{\sigma_s \cup \sigma_{s'}} \subseteq M|_{\sigma_{ss'}}|_{\sigma_s \cup \sigma_{s'}} \)

   (b) \( M_2' \models T_{ss'}^O(p_1) \).
Consider any $\sigma_{ss'}$-structure $M_2$ such that $M_2|_{\sigma_{ss'}} = M'_2$. Then $M_2$ has the same universe $A_2$ as $M'_2$ and

$M_2|_{\sigma \cup \sigma'} = M'_2|_{\sigma \cup \sigma'} \subseteq M_2|_{\sigma_{ss'}}|_{\sigma \cup \sigma'} = M_2|_{\sigma \cup \sigma'}$. Since $M_2|_{\sigma_{ss'}} = M'_2$, $M_2|_{\sigma_{ss'}} \models T_{ss'}^O(p_1)$.

Using Claim 1, then $M_2 \models T_{ss'}^O(p_1)$ and hence $M_2 \models T_{ss'}$.

$|A_1| \leq B_O \leq \max_{O \in \Delta} B_O = B$. 

□
Chapter 10

Deciding The Formulae of InductionSolve

10.1 Special GTSES, Safety Properties and Initial State Conditions

As seen in chapter 8, bounded model property for the formulae $IS_1(k)$ and $IS_2(k)$ allows us to decide their satisfiability. In this section, we show how for particular kinds of graph transforming systems, safety properties and initial state conditions having the bounded model property, we can use their individual bounded model properties to get bounded models for the formulae $IS_1(k)$ and $IS_2(k)$.

We firstly recall, that a GTS $S = (\Delta, \kappa)$ defines a transition relation on states $T(s, s')$ as follows:

Given states $s$ and $s'$, $T(s, s')$ is true iff there exists a $\sigma_{ss'}$—structure $M$ such that $M|_{\sigma_s}$ defines $s$, $M|_{\sigma_{s'}}$ defines $s'$ and $M \models T_{ss'}$.

We again recall that the safety property $P$ and the initial-state conditions $I$ are FO-expressible properties of states. Thus for a state $s$

- $P(s)$ is true iff $s \models \varphi_P$ where $\varphi_P$ is a $\sigma_S$—sentence expressing the safety property.
- $I(s)$ is true iff $s \models \varphi_I$ where $\varphi_I$ is a $\sigma_S$—sentence expressing the initial-state conditions.

We now consider the following special kind of graph-transformation systems, safety properties and initial-state conditions.

Consider the special kind of GTSES $S = (\Delta, \kappa)$ for which $T_{ss'} \in EBS_\Sigma(\sigma)$ where $\Sigma = \sigma_{ss'}$.
and $\sigma = \sigma_s \cup \sigma_{s'}$. Let $B_T$ denote the cardinal satisfying the conditions of the EBS property of $T_{ss'}$.

In the previous chapter, we already saw two kinds of GTSes which have the above property.

Let $\mathcal{Z} = \{ \varphi \mid \varphi \in \text{EBS}_{\Sigma_s}(\Sigma_S) \}$.

Now every formula $\varphi$ can be uniquely encoded as a natural number $\langle \varphi \rangle$. Each of the symbols $\exists, \forall$, the boolean connectives, the parantheses $(, )$, the equality symbol $(=)$, the predicates of $\sigma_S$ can be encoded by assigning unique numbers to them. The separation between adjacent symbols in the formula can be encoded as a unique number representing the separation. The variables can be encoded easily: If there are $n$ variables $x_1, x_2, \ldots, x_n$ appearing in the formula, then we assign a unique number, say $\eta$ for $x_1$ and then the $x_i$ is encoded as the number $\eta \eta \ldots \eta$ times. $\eta$ can always be chosen so that this method of encoding variables doesn’t produce numbers which already represent encodings of other symbols.

For example assign the following codes:

1. Quantifiers: $\exists = 11, \forall = 12$
2. Parenthesis: $( = 21, ) = 22$
3. Boolean connectives: $\wedge = 31, \lor = 32, \neg = 33, \rightarrow = 34, \leftrightarrow = 35$
4. Equality($=$): $= 4$
5. Predicates: $N = 5, E = 6, T_N^j = 7j(1 \leq j \leq N_n), T_E^j = 8j(1 \leq j \leq N_e)$
6. The separator(space) = 0.
7. Variables: If there are $n$ variables $x_1, x_2, \ldots, x_n$, then $x_1$ is encoded as 9, $x_2$ as 99, $x_3$ as 999 and so on

Then $\varphi = \exists x_1 \forall x_2 (N(x_1) \land E(x_1, x_2))$ can be encoded as follows:

$\langle \varphi \rangle = 1109012099021050210902203106021090990220220$.
(Note that using the codes above, we can reconstruct a formula exactly from its encoding.)

Let, for a set $\mathcal{F}$ of formulae, $\langle \mathcal{F} \rangle = \{ \langle \varphi \rangle \mid \varphi \in \mathcal{F} \}$. Then construct the function $f_S : \langle \mathcal{Z} \rangle \to \mathbb{N}$ defined as

$$f_S(\langle \varphi \rangle) = B_\varphi$$

where $B_\varphi$ is the cardinal satisfying the conditions of the EBS property of $\varphi$.

Let $\mathcal{Z}_P$ be a subset of $\mathcal{Z}$ such that

1. It is closed under negation (for every $\varphi \in \mathcal{Z}_P$, $\neg \varphi \in \mathcal{Z}_P$)
2. $f_P = f_S|_{\langle \mathcal{Z}_P \rangle}$ ($f_S$ restricted to $\langle \mathcal{Z}_P \rangle$) is a recursive function.
10.2. **BOUNDED MODELS FOR IS1(k) AND IS2(k)**

(Such a subset exists - Consider \( Z_P \) which contains only 2 sentences of \( Z \) which are negations of each other.) Similarly let \( Z_I \) be a subset of \( Z \) such that

1. It is closed under negation (for every \( \varphi \in Z_I \), \( \neg \varphi \in Z_I \))

2. \( f_I = f_S|_{(Z_I)} \) is a recursive function.

We then consider

- Safety properties \( P \) such that \( \varphi_P \in Z_P \)
- Initial state conditions \( I \) such that \( \varphi_I \in Z_I \)

An example of \( Z_P \) is the following:

\( Z^1_P = \) The set of all sentences which are of the form \( \exists^* \) or \( \forall^* \) in prenex normal form. Since the negation of an \( \exists^* \) sentence is a \( \forall^* \) sentence and vice-versa, \( Z^1_P \) is closed under negation. Further from **Claim 3**, we can see that any \( \exists^* \) sentence and any \( \forall^* \) sentence belongs to \( Z \). Hence \( Z^1_P \subseteq Z \). Further from **Claim 3**, for an \( \exists^* \) sentence \( \varphi \) having \( m \) existential quantifiers, \( B_\varphi = m \) and for a \( \forall^* \) sentence \( \varphi \), \( B_\varphi = 0 \). Then we can construct a procedure for computing \( f_P(n) \).

The procedure takes as input \( n \) and first checks if \( n \) forms a valid encoding. If not, it returns **null**. If yes, then \( n = \langle \varphi \rangle \) for some \( \varphi \). It reconstructs the sentence \( \varphi \) from \( \langle \varphi \rangle \). It brings the sentence in prenex normal form. It then checks whether the sentence is of the form \( \exists^* \) or \( \forall^* \). If its neither of these, it returns **null**. Else if the sentence is of the form \( \exists^* \), then it counts the number(\( m \)) of existential quantifiers and returns \( m \). Else if the sentence is of the form \( \forall^* \), it returns 0.

### 10.2 Bounded Models For IS1(k) and IS2(k)

We recall that

\[
\begin{align*}
IS1(0) &= I(s_1) \land \neg P(s_1) \\
IS2(0) &= \neg P(s_1)
\end{align*}
\]

Consider \( IS1(0) \). Suppose there is a state \( s_1 \) with universe \( A \) such that \( IS1(0) \) is true. Then

\[ s_1 \models \varphi_I \land \neg \varphi_P \]
Then, from our assumptions about $\varphi_I$, there exists a subset $A_1 \subseteq A$ such that $|A_1| \leq f_1((\varphi_I))$ satisfying the properties mentioned. Similarly for $\neg \varphi_P$, there exists a subset $A_2 \subseteq A$ such that $|A_2| \leq f_P((\neg \varphi_P))$ and satisfying the properties mentioned. Construct

$$A' = A_1 \cup A_2$$

$$|A'| \leq (|A_1| + |A_2|) \leq (f_1(\neg \varphi_I)) + f_P((\neg \varphi_P)).$$

Consider the substructure $s'_1$ of $s$ generated by $A'$. Then from assumptions about $\varphi_I$ and $\varphi_P$,

$$s'_1 \models \varphi_I \land \neg \varphi_P$$

Thus given $s_1$ satisfying $IS1(0)$, there is a state $s'_1$ of bounded size ($|A'| \leq (f_1(\neg \varphi_I)) + f_P((\neg \varphi_P))$) satisfying $IS1(0)$.

We can similarly show that, given $s_1$ satisfying $IS2(0)$, there is a state $s'_1$ of bounded size ($|A'| \leq f_P((\neg \varphi_P))$) satisfying $IS2(0)$.

Let us consider $IS1(k)$ and $IS2(k)$ for $k \geq 1$. We have,

$$IS1(k) = \bigwedge_{i=1}^{i=k} T(s_i, s_{i+1}) \land \bigwedge_{i=1}^{i=k} P(s_i) \land \bigwedge_{i=2}^{i=k+1} \neg I(s_i) \land \bigwedge_{1 \leq i,j \leq k; i \neq j} (s_i \neq s_j) \land I(s_1) \land \neg P(s_{k+1})$$

$$IS2(k) = \bigwedge_{i=1}^{i=k} T(s_i, s_{i+1}) \land \bigwedge_{i=1}^{i=k} P(s_i) \land \bigwedge_{1 \leq i,j \leq k; i \neq j} (s_i \neq s_j) \land \neg P(s_{k+1})$$

We show now that if there exists a sequence of states $s_1, s_2, \ldots, s_{k+1}$ satisfying $IS1(k)$ (resp. $IS2(k)$), there exists a sequence of bounded states $s'_1, s'_2, \ldots, s'_{k+1}$ satisfying $IS1(k)$ (resp. $IS2(k)$). We note that since in InductionSolve, we proceed inductively i.e. checking $IS1(k)$ and $IS2(k)$ starting with $k = 0$ and increasing its value, while considering $k \geq 1$, we have already established that $I(s) \rightarrow P(s)$ (and hence $\neg P(s) \rightarrow \neg I(s)$) for any state $s$.

Consider a sequence of states $s_1, s_2, \ldots, s_{k+1}$ satisfying $IS1(k)$.

Since $\neg P(s_{k+1})$ is true, $s_{k+1} \models \neg \varphi_P$. Let the universe of $s_{k+1}$ be $A$. By assumption about $\neg \varphi_P$, there exists a subset $A^1_{k+1} \subseteq A$ such that $|A^1_{k+1}| \leq f_P((\neg \varphi_P))$ satisfying the properties mentioned. Let $A_{k+1} = A^1_{k+1}$.

Now consider $s_k$ and $s_{k+1}$. Since $T(s_k, s_{k+1})$ is true, there exists a $\sigma_{ss'}$—structure $M_k$ such that $M_k|\sigma$ defines $s_k$, $M_k|\sigma_s$ defines $s_{k+1}$ and $M_k \models T_{ss'}$. (Thus $M_k, s_k, s_{k+1}$ have the same universe, namely $A$.) Then by assumption, since $T_{ss'} \in \text{EBS}_\Sigma(\Sigma = \sigma_{ss'}, \sigma = \sigma_s \cup \sigma_{s'})$, there exists $A^1_k \subseteq A$ such that $|A^1_k| \leq B_T$ and which satisfies the conditions of the EBS property.

Since $P(s_k)$ is true, $s_k \models \varphi_P$ and hence by assumption, there is a subset $A^2_k \subseteq A$ such that $|A^2_k| \leq f_P(\langle \varphi_P \rangle)$ and which satisfies the properties of $Z_P$. 

Further since $\neg \mathcal{I}(s_k)$ is true, $s_k \models \neg \varphi_I$ and hence by assumption, there is a subset $A^3_k \subseteq A$ such that $|A^3_k| \leq f_I(\langle \neg \varphi_I \rangle)$ and which satisfies the properties of $Z_I$.

Construct

$$A_k = A^1_k \cup A^2_k \cup A^3_k \cup A_{k+1}$$

Note that

1. $A_{k+1} \subseteq A_k \subseteq A$

2. $|A_k| \leq |A^1_k| + |A^2_k| + |A^3_k| + |A_{k+1}|

$$\leq B_T + f_P(\langle \varphi_P \rangle) + f_I(\langle \neg \varphi_I \rangle) + |A_{k+1}|$$

Now consider $s_{k-1}$ and $s_k$. Since $\mathcal{I}(s_{k-1}, s_k)$ is true, there exists a $\sigma_{ss'}$-structure $M_{k-1}$ such that $M_{k-1}|\sigma_s$ defines $s_{k-1}$, $M_{k-1}|\sigma_s'$ defines $s_k$ and $M_{k-1} \models T_{ss'}$. (Thus $M_{k-1}, s_{k-1}, s_k$ have the same universe, namely A.) Then by assumptions, since $T_{ss'} \in \text{EBS}_\Sigma(\Sigma = \sigma_{ss'}, \sigma = \sigma_s \cup \sigma_s')$, there exists $A^1_{k-1} \subseteq A$ such that $|A^1_{k-1}| \leq B_T$ satisfying the conditions of the EBS property.

Since $\mathcal{P}(s_{k-1})$ is true, $s_{k-1} \models \varphi_P$ and hence by assumption, there is a subset $A^2_{k-1} \subseteq A$ such that $|A^2_{k-1}| \leq f_P(\langle \varphi_P \rangle)$ and which satisfies the properties of $Z_P$.

Further since $\neg \mathcal{I}(s_{k-1})$ is true, $s_{k-1} \models \neg \varphi_I$ and hence by assumption, there is a subset $A^3_{k-1} \subseteq A$ such that $|A^3_{k-1}| \leq f_I(\langle \neg \varphi_I \rangle)$ and which satisfies the properties of $Z_I$.

Construct

$$A_{k-1} = A^1_{k-1} \cup A^2_{k-1} \cup A^3_{k-1} \cup A_k$$

Note that

1. $A_k \subseteq A_{k-1} \subseteq A$

2. $|A_{k-1}| \leq |A^1_{k-1}| + |A^2_{k-1}| + |A^3_{k-1}| + |A_k|

$$\leq B_T + f_P(\langle \varphi_P \rangle) + f_I(\langle \neg \varphi_I \rangle) + |A_k|$$

We continue this way back to $s_1$. At $s_1$, we construct $A^1_1$ as we constructed any $A^1_i$.

Now since $\mathcal{I}(s_1)$ is true, $s_1 \models \varphi_I$ and hence by assumption, there is a subset $A^2_1 \subseteq A$ such that $|A^2_1| \leq f_I(\langle \varphi_I \rangle)$ and which satisfies the properties of $Z_I$.

Then construct

$$A_1 = A^1_1 \cup A^2_1 \cup A_2$$

Note that

1. $A_2 \subseteq A_1 \subseteq A$

2. $|A_1| \leq |A^1_1| + |A^2_1| + |A_2|

$$\leq B_T + f_P(\langle \varphi_P \rangle) + |A_2|$$

(Note that since, we have already established that $\mathcal{I}(s) \rightarrow \mathcal{P}(s)$, we do not consider the $f_P(\langle \varphi_P \rangle)$ term for $|A_1|$. By the same reason, we do not consider the $f_I(\langle \neg \varphi_I \rangle)$ term for
Thus we have a sequence of $A_i$s satisfying the following recurrence relations

1. $A_i = A_1^i \cup A_2^i \cup A_3^i \cup A_{i+1}$ \hspace{1em} $1 \leq i \leq k$
2. $A_{i+1} \subseteq A_i \subseteq A$ \hspace{1em} $1 \leq i \leq k$
3. $|A_i| \leq B_T + f_P(\langle \varphi_P \rangle) + f_I(\langle \neg \varphi_I \rangle) + |A_{i+1}|$ \hspace{1em} $2 \leq i \leq k$
4. $|A_k+1| \leq f_P(\langle \neg \varphi_P \rangle)$
5. $|A_1| \leq B_T + f_I(\langle \varphi_I \rangle)$

Consider $s_i$ and $s_j$ ($1 \leq i, j \leq k+1, i \neq j$). $s_i$ and $s_j$ have the same universe, namely $A$. Now since $s_i \neq s_j$, there exists $A_{ij} \subseteq A$ such that the substructure $s_i^1$ of $s_i$ generated by $A_{ij}$ is not equal to the substructure $s_j^1$ of $s_j$ generated by $A_{ij}$, i.e. $s_i^1 \neq s_j^1$.

Consider $A_{ij}^*$ such that

$$|A_{ij}^*| = \min \{|A_{ij}| \mid A_{ij} \subseteq A, s_i^1 \neq s_j^1\}$$

Since $\sigma_S$ consists of only unary and binary predicates, we can see that

$$|A_{ij}^*| \leq 2 \hspace{1em} 1 \leq i, j \leq k+1, i \neq j$$

Construct

$$A' = A_1 \cup \bigcup_{1 \leq i, j \leq k+1, i \neq j} A_{ij}^*$$

Size Calculation for $A'$

Solving the recurrence relation

$$|A_i| \leq B_T + f_P(\langle \varphi_P \rangle) + f_I(\langle \neg \varphi_I \rangle) + |A_{i+1}| \hspace{1em} 2 \leq i \leq k$$
$$|A_{k+1}| = f_P(\langle \neg \varphi_P \rangle)$$

we get

$$|A_i| \leq (k + 1 - i)(B_T + f_P(\langle \varphi_P \rangle) + f_I(\langle \neg \varphi_I \rangle)) + f_P(\langle \neg \varphi_P \rangle)$$

Then

$$|A'| \leq |A_1| + \sum_{1 \leq i, j \leq k+1, i \neq j} |A_{ij}^*|$$
$$\leq B_T + f_I(\langle \varphi_I \rangle) + |A_2| + \sum_{1 \leq i, j \leq k+1, i \neq j} 2$$
$$\leq kB_T + (k - 1)(f_P(\langle \varphi_P \rangle) + f_I(\langle \neg \varphi_I \rangle)) + f_I(\langle \varphi_I \rangle) + f_P(\langle \neg \varphi_P \rangle) + 2^{k+1}C_2$$
Thus $A'$ is of bounded size.

Now for each $i, 1 \leq i \leq k + 1$, consider $M_i$ (seen earlier) such that $M_i \models T_{ss'}$ and $M_i|_{\sigma_s}$ defines $s_i$ and $M_i|_{\sigma_s'}$ defines $s_{i+1}$. $M_i$ has universe $A$. Consider $A^1_i$. Since $A^1_i \subseteq A' \subseteq A$, from the assumptions about the GTS, there exists a $\sigma_{ss'}$-structure $M^1_i$ such that

1. $M^1_i|_{\sigma_{s} \cup \sigma_{s'}} \subseteq M_i|_{\sigma_{s} \cup \sigma_{s'}}$

2. $M^1_i \models T_{ss'}$

Consider $s'_i$ and $s'_{i+1}$ such that $M^1_i|_{\sigma_s}$ defines $s'_i$ and $M^1_i|_{\sigma_{s'}}$ defines $s'_{i+1}$.

1. Since $M^1_i \models T_{ss'}$, $T(s'_i, s'_{i+1})$ is true.

2. Since $M^1_i|_{\sigma_{s} \cup \sigma_{s'}} \subseteq M_i|_{\sigma_{s} \cup \sigma_{s'}}$, $M^1_i|_{\sigma_{s}} \subseteq M_i|_{\sigma_{s}}$. Thus $s'_i \subseteq s_i$. Similarly $s'_{i+1} \subseteq s_{i+1}$.

3. For $2 \leq i \leq k$, $P(s_i)$ is true, hence $s_i \models \varphi_P$. Further $A^2_i \subseteq A_i \subseteq A' \subseteq A$ and $s'_i$ (having universe $A'$) is a substructure of $s_i$ (having universe $A$). Then from assumptions about $\varphi_P$, $s'_i \models \varphi_P$. Hence $P(s'_i)$ is true.

4. For $2 \leq i \leq k$, $\neg I(s_i)$ is true, hence $s_i \models \neg \varphi_I$. Further $A^3_i \subseteq A_i \subseteq A' \subseteq A$ and $s'_i$ (having universe $A'$) is a substructure of $s_i$ (having universe $A$). Then from assumptions about $\neg \varphi_I$, $s'_i \models \neg \varphi_I$. Hence $\neg I(s'_i)$ is true.

5. Consider $i = k + 1$. Since $\neg P(s_{k+1})$ is true, $s_{k+1} \models \neg \varphi_P$. Further $A_{k+1} \subseteq A_1 \subseteq A' \subseteq A$ and $s'_{k+1}$ (having universe $A'$) is a substructure of $s_i$ (having universe $A$). Then from assumptions about $\neg \varphi_P$, $s'_{k+1} \models \neg \varphi_P$. Hence $\neg P(s'_{k+1})$ is true.

6. Consider $i = 1$. Since $I(s_1)$ is true, $s_1 \models \varphi_I$. Further $A^2_1 \subseteq A_1 \subseteq A' \subseteq A$ and $s'_1$ (having universe $A'$) is a substructure of $s_i$ (having universe $A$). Then from assumptions about $\varphi_I$, $s'_1 \models \varphi_I$. Hence $I(s'_1)$ is true.

Since we already established that $I(s) \rightarrow P(s)$, we have

1. $I(s'_1) \rightarrow P(s'_1)$ and hence $P(s'_1)$ is true.

2. $\neg P(s'_{k+1}) \rightarrow \neg I(s'_{k+1})$ and hence $\neg I(s'_{k+1})$ is true.

Finally for $1 \leq i, j \leq k + 1, i \neq j$, $A^*_{ij}$ is such that the substructure $s'_i$ of $s_i$ generated by $A^*_{ij}$ and the substructure $s'_j$ of $s_j$ generated by $A^*_{ij}$ are such that $s'_i \neq s'_j$. Now since $A^*_{ij} \subseteq A' \subseteq A$, $s'_i \subseteq s'_i$ and $s'_j \subseteq s'_j$. Since $s'_i \neq s'_j$, we have $s'_i \neq s'_j$.

Thus we have shown that if there is a sequence of states $s_1, s_2, \ldots, s_{k+1}$ satisfying $IS1(k)$,
there exists a sequence of states \( s'_1, s'_2, \ldots, s'_{k+1} \) satisfying \( IS1(k) \) such that each \( s'_i \) is bounded. Each \( s'_i \) has universe \( A' \) which is bounded by \( B \) given by

\[
B = k.B_T + (k-1).(f_P(\langle \varphi_P \rangle) + f_I(\langle \neg \varphi_I \rangle) + f_I(\langle \varphi_I \rangle) + f_P(\langle \neg \varphi_P \rangle) + 2.\kappa^2 + C_2
\]

It can be shown that the \( 2.\kappa^2 + C_2 \) term can be replaced by a \( 2.(k - 1) \) term so that the bound is given by

\[
B = k.B_T + (k-1).(f_P(\langle \varphi_P \rangle) + f_I(\langle \neg \varphi_I \rangle) + f_I(\langle \varphi_I \rangle) + f_P(\langle \neg \varphi_P \rangle))
\]

In a very similar fashion, we can show that if there is a sequence of states \( s_1, s_2, \ldots, s_{k+1} \) satisfying \( IS2(k) \), there exists a sequence of states \( s'_1, s'_2, \ldots, s'_{k+1} \) satisfying \( IS2(k) \) such that each \( s'_i \) is bounded. Each \( s'_i \) has universe \( A' \) which is bounded by \( B \) given by

\[
B \leq k.(B_T + f_P(\langle \varphi_P \rangle)) + f_P(\langle \neg \varphi_P \rangle)
\]

Thus for \( IS1(k) \) (resp. \( IS2(k) \)), if we can find \( s'_1, s'_2, \ldots, s'_{k+1} \) having universes of size bounded (by the bound we calculated above) satisfying \( IS1(k) \) (resp. \( IS2(k) \)), then we have found a sequence of states satisfying \( IS1(k) \) (resp. \( IS2(k) \)). If however, we cannot find \( s'_1, s'_2, \ldots, s'_{k+1} \) having universes of size bounded (by the bound we calculated above) satisfying \( IS1(k) \) (resp. \( IS2(k) \)), then there cannot exist a sequence of states \( s_1, s_2, \ldots, s_{k+1} \) satisfying \( IS1(k) \) (resp. \( IS2(k) \)), since if so, we should have been able to find \( s'_1, s'_2, \ldots, s'_{k+1} \) having universes of bounded size satisfying \( IS1(k) \) (resp. \( IS2(k) \)).

### 10.3 Decision Procedure For InductionSolve

Using the results of the previous section, we can now construct a decision procedure for checking the validity of the formulae appearing in InductionSolve. We denote the formulae of InductionSolve by \( ISF1(k) \) and \( ISF2(k) \) where \( ISF1(k) = \neg IS1(k) \) and \( ISF2(k) = \neg IS2(k) \).

The procedure takes in as input the sentences \( T_{ss'}, \varphi_P \) and \( \varphi_I, k \) (value of iterator in InductionSolve) and \( J (J = 1, 2 \) which indicates which formula to decide i.e. \( ISF1 \) or \( ISF2 \)) as input. It is assumed that \( B_T \) and the functions \( f_I \) and \( f_P \) are known. The procedure then performs the following operations:

1. Formula Generation
   
   a. For each \( i, 1 \leq i \leq k \), it produces the sentence \( T_{s_i,s_{i+1}} \) as follows:
      
      i. It replaces each occurrence of the predicates \( N_s \) and \( N_{s'} \) with \( N_{s_i} \) and \( N_{s_{i+1}} \).
      
      It does likewise for predicates \( E_s, E_{s'}, T^j_{N,s}, T^j_{N,s'} (1 \leq j \leq N_e), T^j_{E,s}, T^j_{E,s'} (1 \leq j \leq N_e). \)
ii. For each operation \( O = (\Omega, p) \in \Delta \) (where the GTS is \( S = (\Delta, \kappa) \)), for each \( \xi \in \Omega \), it replaces each occurrence of

A. \( X^x_n \) with \( X^x{\lambda}_n^s_{i_1} \), \( \lambda_n \in \kappa_N \)
B. \( Y^x_e \) with \( Y^x{\mu}_e^s_{i_1} \), \( \mu_e \in \kappa_E \)
C. \( X^O_n \) with \( X^O{\lambda}_n^s_{i_1} \), \( \lambda_n \in \kappa_N \)
D. \( Y^O_e \) with \( Y^O{\mu}_e^s_{i_1} \), \( \mu_e \in \kappa_E \)

(b) For each \( i, 1 \leq i \leq k+1 \), it produces the sentence \( \varphi_{P, s_i} \) from \( \varphi_P \) by replacing each occurrence of the predicate \( N \) with \( N_{s_i} \), \( E \) with \( E_{s_i} \), \( T_N^1 \) with \( T_N^{s_i} \)(1 \( \leq j \leq N_n \)), and \( T_E^1 \) with \( T_E^{s_i} \)(1 \( \leq j \leq N_e \)).

(c) For each \( i, 1 \leq i \leq k + 1 \), it produces the sentence \( \varphi_{I, s_i} \) from \( \varphi_I \) as in (2).

(d) For \( 1 \leq i, j \leq k + 1, i \neq j \), it produces the sentence

\[
\text{Neq}_{s_i, s_j} = \neg \left( \forall x \left( N_{s_i}(x) \leftrightarrow N_{s_j}(x) \right) \land \forall x, y \left( E_{s_i}(x, y) \leftrightarrow E_{s_j}(x, y) \right) \right)
\]

(e) • If \( J = 1 \), it produces the sentence

\[
\text{IS}_{1k} = \bigwedge_{i=1}^{i=k} \tau_{s_i, s_i+1} \land \bigwedge_{i=1}^{i=k} \varphi_{P, s_i} \land \bigwedge_{i=2}^{i=k+1} \neg \varphi_{I, s_i} \land \bigwedge_{1 \leq i, j \leq k+1; i \neq j} \text{Neq}_{s_i, s_j} \land \varphi_{I, s_i} \land \neg \varphi_{P, s_{k+1}}
\]

• Else if \( J = 2 \), it produces the sentence

\[
\text{IS}_{2k} = \bigwedge_{i=1}^{i=k} \tau_{s_i, s_i+1} \land \bigwedge_{i=1}^{i=k} \varphi_{P, s_i} \land \bigwedge_{1 \leq i, j \leq k+1; i \neq j} \text{Neq}_{s_i, s_j} \land \neg \varphi_{P, s_{k+1}}
\]

2. Removal of equality.

It then removes equality (\( = \)) between variables as follows:

Let \( \sigma \) be the set of all predicates appearing in \( ISJ_k \). Let \( P \) be a \( k \)-ary predicate in \( \sigma \). Then \( x = y \) is replaced with

\[
\bigwedge_{P \in \sigma} \forall z_1, z_2, \ldots, z_{k-1} \left[
\begin{array}{l}
P(x, z_1, z_2, \ldots, z_{k-1}) \leftrightarrow P(y, z_1, z_2, \ldots, z_{k-1}) \\
P(z_1, x, z_2, \ldots, z_{k-1}) \leftrightarrow P(z_1, y, z_2, \ldots, z_{k-1}) \\
\vdots \\
P(z_1, z_2, \ldots, z_{k-1}, x) \leftrightarrow P(z_1, z_2, \ldots, z_{k-1}, y)
\end{array}\right]
\]

Let \( ISJ^1_k \) be the resulting formula after removal of equalities.
   It removes negations just preceding quantifiers using the rules \( \neg \exists = \forall \neg \) and \( \neg \forall = \exists \neg \) to produce \( ISJ_k^2 \).

4. Bound computation.
   It then computes the bound \( B \) for \( ISJ(k) \) using the expression derived earlier. Since \( B_T \) is known and since \( f_P \) and \( f_I \) are recursive functions, it computes \( B \) by using the procedures for these. It then constructs the set \( A = \{ a_1, a_2, \ldots, a_B \} \).

5. Elimination of quantifiers.
   It eliminates quantifiers.
   - Each occurrence of \( \exists x \) is replaced with \( \bigvee_{x \in A} x \).
   - Each occurrence of \( \forall x \) is replaced with \( \bigwedge_{x \in A} x \).

   Let the resulting formula be \( ISJ_k^3 \).
   Note that after this step there are no quantifiers or variables in \( ISJ_k^3 \).

6. Propositionalization.
   It propositionalises \( ISJ_k^3 \) to produce \( ISJ_k^4 \).
   For each \( P \in \sigma \), \( P \)-ary, it replaces \( P(a_{i_1}, a_{i_2}, \ldots, a_{i_k})(1 \leq i_j \leq B, 1 \leq j \leq k) \) with the propositional variable \( P_{a_{i_1}, \ldots, a_{i_k}} \).

7. Invoking a propositional SAT solver.
   The formula \( ISJ_k^4 \) is a propositional logic formula and is fed to a propositional SAT solver which decides the satisfiability of the formula.

8. Output.
   If the SAT solver returns \textbf{True}, it means \( ISJ_k^4 \) and hence \( ISJ_k \) is satisfiable. Then \( ISFJ(k) \) is not valid and hence the procedure outputs \textbf{False}. The satisfying assignment produced by the SAT solver gives the counterexample for \( ISFJ(k) \).
   If the SAT solver returns \textbf{False}, it means \( ISJ_k^4 \) and hence \( ISJ_k \) is unsatisfiable. Then \( ISFJ(k) \) is valid and hence the procedure outputs \textbf{True}.

The entire flow is shown in the figure below.
10.4 Checking the Library System

We have already shown in chapter 9 that the library system can be expressed as a GTS which is of the kind mentioned in the previous chapter. We now look at the safety property and the initial state conditions.

The library system starts from the empty graph. Thus $I$ is expressed by the formula $\varphi_I$ which is given as

$$\varphi_I = \forall x \neg N(x) \land \forall x, y, \neg E(x, y)$$

$$= \forall x, y (\neg N(x) \land \neg E(x, y))$$

Thus $\varphi_I$ is a $\forall^*$ formula. Thus $\varphi_I$ belongs to the example $Z^1_P$, which we had considered in 10.1.

The safety property $\mathcal{P}$ we wanted to check was the $C-T-B-S$ pattern. This pattern is present in a state if there is a Claim instance connected to a Title instance which is connected to a Book instance which in turn is connected to a Shelf instance. If this pattern is present it means that a claim for a title is kept pending even when there is a book for the title on shelf which is clearly an undesirable situation. We can express this as a safety property by the formula $\varphi_P$ which is given as

$$\varphi_P =$$

$$\neg \exists c, t, b, s (C(c) \land T(t) \land B(b) \land S(s) \land N(c) \land N(t) \land N(b) \land N(s) \land E(c, t) \land E(b, t) \land E(b, s))$$

$$\forall c, t, b, s (C(c) \land T(t) \land B(b) \land S(s) \land N(c) \land N(t) \land N(b) \land N(s) \land E(c, t) \land E(b, t) \land E(b, s))$$

Thus $\varphi_P$ is also a $\forall^*$ formula. Thus $\varphi_P$ also belongs to the example $Z^1_P$, which we had considered in 10.1.

We just proved that for the kind of GTSes, safety properties and initial state conditions mentioned in 10.1, we can decide the formulae IS1($k$) and IS2($k$). The library system, the $C-T-B-S$ safety property and the initial graph of the library system - are all of the kind mentioned in 10.1. Thus we can decide the formulae IS1($k$) and IS2($k$) for the library system.

For the $C-T-B-S$ safety property, $f_P(\langle \varphi_P \rangle) = 0$ and $f_P(\langle \neg \varphi_P \rangle) = 4$. For the initial state, $f_I(\langle \varphi_I \rangle) = 0$ and $f_I(\langle \neg \varphi_I \rangle) = 2$. For the library system, $B_T = 8$. The bounds for $k \geq 1$ for the system were - (a) $IS1(k) : B = 12k - 1$ (b) $IS2(k) : B = 8k + 3$. Though in general we cannot put bounds on the iterator $k$ of InductionSolve, it turns out that for $k = 1$ itself, InductionSolve terminates for the library system returning True, meaning that the library system does satisfy the $C-T-B-S$ safety property.
CHAPTER 10. DECIDING THE FORMULAE OF INDUCTION SOLVE

Validity Checker

1. Removal of equality
2. Removal of negations preceding quantifiers.
4. Elimination of quantifiers.
5. Propositionalisation.
Chapter 11

Some studies on the EBS property

11.1 A Syntactic Subclass of EBS

(Note: There is a modification to the definition of the QPS property and a slight modification to Example 1, and the statements of Lemmas 4 and 5 as given in the paper (submission no: 139) submitted to FSTTCS ‘07. These should be as given below.)

Without loss of generality, consider a formula \( \varphi(x) \) in prenex normal form with all variables named uniquely. Let \( V(\varphi(x)) \) be the set of free variables and variables that are existentially quantified before (to the left of) all universally quantified variables in the prefix structure of \( \varphi(x) \). Let \( AV(\varphi(x)) \) be the set of all \( \forall \) quantified variables of \( \varphi(x) \) and \( EV(\varphi(x)) \) be the set of all \( \exists \) quantified variables of \( \varphi(x) \) not in \( V(\varphi(x)) \). Further, we index the quantified variables in the prefix in increasing order from left to right.

Definition 11.1 (QPS property) The formula \( \varphi(x) \) is said to satisfy the Quantifier based Predicate Separable property with respect to \( \sigma \subseteq \Sigma \) (denoted as \( \varphi(x) \in QPS_{\Sigma}(\sigma) \)) if \( \text{Sig}(\varphi) = \Sigma \) and the following conditions hold:

1. Every predicate \( P \in \Sigma \) of arity \( k \geq 1 \) satisfies the following:
   
   - If \( P \in \sigma \), then every argument of every instance of \( P \) is in \( V(\varphi(x)) \cup AV(\varphi(x)) \).
   - If \( P \in \Sigma \setminus \sigma \), then for each \( i \) in 1 through \( k \), either (a) the \( i^{th} \) argument of every instance of \( P \) in \( \varphi \) is in \( V(\varphi(x)) \cup AV(\varphi(x)) \), or (b) the \( i^{th} \) argument of every instance of \( P \) in \( \varphi \) is in \( V(\varphi(x)) \cup EV(\varphi(x)) \). Further, for each instance of \( P \), either (a) all its arguments are from \( V(\varphi(x)) \cup AV(\varphi(x)) \) or (b) its highest indexed argument is from \( EV(\varphi(x)) \).

2. For any 2 instances of \( P \in \Sigma \setminus \sigma \) of arity \( \geq 1 \) having non-indentical argument lists, either (a) all arguments of each instance are from \( V(\varphi(x)) \cup AV(\varphi(x)) \) or (b) there exists a variable from \( EV(\varphi(x)) \) which appears as the \( i^{th} \) argument of one instance but not as the \( i^{th} \) argument of the other.

83
Example 1:
In the formula $\varphi(u)$ as given in the paper, the $\neg Q(z)$ should actually be $\neg Q(x)$. The rest of the formula remains unchanged.

Lemma 4:
Let $\varphi(x)$ be a formula in prenex normal form having the $\exists^* \forall^* \exists^*$ quantifier prefix, in which every predicate of arity $\geq 1$ in $\Sigma = \text{Sig}(\varphi)$ appears exactly once. Then there exists $\sigma \subseteq \Sigma$ such that $\varphi(x) \in \text{QPS}_\Sigma(\sigma)$.

Proof: Follows from definition.

Lemma 5:
Let $\varphi(x)$ be a formula in QPS$_\Sigma(\sigma)$ in prenex normal form. Let $k$ be the total number of instances of all predicates of arity $\geq 1$ in $\varphi(x)$ and $k_0$ be the total number of nullary predicate instances in $\varphi(x)$. Let $r = |\text{EV}(\varphi(x))|$. Then $\varphi(x)$ satisfies all conditions in EBS Definition with $B = k_0 + |x| + r.3^k$.

Proof:
Consider a $\Sigma$-structure $M$ such that $M \models \varphi(b_1)$ and such that $|A| \geq B = k_0 + |x| + r.3^k$. Let $b_2$ be the witnesses to the leftmost $\exists$ quantified variables of $\varphi$. Consider any subset $A_1$ of $A$ such that $b_1 \in A_1[x], b_2 \in A_1^{[b_2]}$ and such that in addition to these elements (i.e. different from these) $A_1$ has $r.3^k$ distinct elements of $A$. Then $|A_1| \leq B$. Let the $r.3^k$ elements be denoted as $a_{(1,1)}, \ldots, a_{(1,3^k)}, a_{(2,1)}, \ldots, a_{(2,3^k)}, \ldots, a_{(r,1)}, \ldots, a_{(r,3^k)}$. Consider any $A_2$ such that $A_1 \subseteq A_2 \subseteq A$.

Let $\varphi(x)$ be of the form

$$\varphi(x) = \exists y \forall z_0 \exists v_1 \forall z_1 \exists v_2 \forall z_2 \ldots \exists v_r \forall z_r \psi(x, y, z_0, v_1, z_1, \ldots, v_r, z_r)$$

where the matrix $\psi$ is quantifier-free. We now show how to choose $v_1, \ldots, v_r$ and how to define the predicates of $\Sigma \setminus \sigma$ so that we can get the desired $\Sigma$-structure $M_2$ satisfying the EBS conditions. Choose $x = b_1$ and $y = b_2$. Recall that $A_2 \subseteq A$.

We move from left to right in the quantifier prefix of $\varphi$ and observe the truth values in $M$ of the predicate instances of $\psi$ as we substitute values from $A_2$ for the $\forall$ quantified variables and values from $A$ for the $\exists$ quantified variables. First we substitute the values for the variables of $V(\varphi(x))$, namely $b_1$ for $x$ and $b_2$ for $y$. Next we substitute values for $z_0$ from $A_2$. For each value $d_0 \in A_2^{[z_0]}$ of $z_0$, there is a certain value say $e_1$ of $v_1$ ($e_1 \in A$) which is used to make $\varphi(b_1)$ true in $M$. Upon substitution of $e_1$ for $v_1$ in $\psi$, each predicate instance when interpreted in $M$ with the (possibly incompletely specified) values of its arguments as they appear in $\psi(b_1, b_2, d_0, e_1, z_1, \ldots, v_r, z_r)$, can be in one of 3 ‘states’: (a) True (b) False (c) Undetermined at that stage (because the values of the subsequent variables in the prefix sequence of $\varphi$ have yet not been specified.) Since each predicate instance is in one of these 3 states, the states of all predicate instances taken together determines a ‘configuration’ of states. Since there are $k$ predicate instances, there are $3^k$ configurations. Number the $3^k$ configurations from 1 to $3^k$. Then upon substitution of value $e_1$ for $v_1$, let
11.1. A SYNTACTIC SUBCLASS OF EBS

Let $j_1$ be the configuration of the states of all predicates instances, when interpreted in $M$, for the values of their arguments as they appear in $\psi(b_1, b_2, d_0, e_1, z_1, \ldots, v_r, z_r)$. Then from $A_2$, we choose the value $a_{(1,j_1)}$ to substitute for $v_1$. Further we define the interpretations in $M_2$ of the predicates of $\Sigma$ such that the states of the predicate instances, when interpreted in $M_2$, for the values of their arguments as they appear in $\psi(b_1, b_2, d_0, a_{(1,j_1)}, z_1, \ldots, v_r, z_r)$, produce the same configuration, namely $j_1$. In particular, each predicate instance, when interpreted in $M_2$, with the (possibly incompletely specified) values of its arguments as they appear in $\psi(b_1, b_2, d_0, a_{(1,j_1)}, z_1, \ldots, v_r, z_r)$, has the same state as it has when interpreted in $M$ with the (possibly incompletely specified) values of its arguments as they appear in $\psi(b_1, b_2, d_0, e_1, z_1, \ldots, v_r, z_r)$.

Next we substitute values for $z_1$ from $A_2$. For each value $d_1 \in A_2^{z_1}$ of $z_1$, there is a certain value say $e_2$ of $v_2$ ($e_2 \in A$) which is used to make $\varphi(b_1)$ true in $M$. Upon substitution of value $e_2$ for $v_2$, let $j_2$ be the configuration of the states of all predicates instances, when interpreted in $M$, for the values of their arguments as they appear in $\psi(b_1, b_2, d_0, e_1, d_1, e_2, z_2, \ldots, v_r, z_r)$. Then from $A_2$, we choose the value $a_{(2,j_2)}$ to substitute for $v_2$. Further we define the interpretations in $M_2$ of the predicates of $\Sigma$ such that the states of the predicate instances, when interpreted in $M_2$, for the values of their arguments as they appear in $\psi(b_1, b_2, d_0, a_{(1,j_1)}, d_1, a_{(2,j_2)}, z_2, \ldots, v_r, z_r)$, produce the same configuration, namely $j_1$. In particular, each predicate instance, when interpreted in $M_2$, with the (possibly incompletely specified) values of its arguments as they appear in $\psi(b_1, b_2, d_0, a_{(1,j_1)}, d_1, a_{(2,j_2)}, z_2, \ldots, v_r, z_r)$ has the same state as it has when interpreted in $M$ with the (possibly incompletely specified) values of its arguments as they appear in $\psi(b_1, b_2, d_0, e_1, d_1, e_2, z_2, \ldots, v_r, z_r)$.

We proceed this way till eventually for each $d_0 \in A_2^{z_0}, \ldots, d_r \in A_2^{z_r}$, we would have determined the values $a_{i,j_i}$ for $v_i (1 \leq i \leq r)$ and would have defined the state of each predicate instance, when interpreted in $M_2$, for the values of its arguments as they appear in $\psi(b_1, b_2, d_0, a_{(1,j_1)}, d_1, a_{(2,j_2)}, \ldots, a_{(r,j_r)}, d_r)$. In particular, since in $\psi(b_1, b_2, d_0, a_{(1,j_1)}, d_1, a_{(2,j_2)}, \ldots, a_{(r,j_r)}, d_r)$ all predicate instances have values for all arguments specified, we would have defined the truth values in $M_2$ of each predicate instance, for the values of its arguments as they appear in $\psi(b_1, b_2, d_0, a_{(1,j_1)}, d_1, a_{(2,j_2)}, \ldots, a_{(r,j_r)}, d_r)$. Note that if for a predicate instance, all arguments are from $V(\varphi(x)) \cup AV(\varphi(x))$, then its interpretation in $M_2$ is exactly the same as in $M$ restricted to $A_2$.

A conflict is said to occur for a predicate $P$ if for some set of values of its arguments, it has been defined to be both True and False. The question that now arises is: Does the interpretation of the predicates in $M_2$, the way we have constructed above, contain a conflict for any predicate?

Now a conflict can occur in two ways:

1. Internal: Given values $d_0, d_1, \ldots, d_r$, there are 2 different instances $I_1$ and $I_2$ of the same predicate which have the same set of values for their arguments in $\psi(b_1, b_2, d_0, a_{(1,j_1)}, d_1, a_{(2,j_2)}, \ldots, a_{(r,j_r)}, d_r)$ but (w.l.o.g) evaluate to True and False.
respectively, when interpreted in $M_2$.

2. External: Given values $d_{01}, d_{11}, \ldots, d_{i1}$ and $d_{02}, d_{12}, \ldots, d_{r2}$ (and hence the corresponding values $a_{(11,j_{11})}, \ldots, a_{(r1,j_{r1})}$ and $a_{(12,j_{12})}, \ldots, a_{(r2,j_{r2})}$ to the variables $v_1, \ldots, v_r$), there are 2 instances $I_1$ and $I_2$ of the same predicate such that $I_1$ has the same set of values for its arguments in $\psi(b_1, b_2, d_{01}, a_{(11,j_{11})}, d_{11}, a_{(21,j_{21})}, \ldots, a_{(r1,j_{r1})}, d_{r1})$ as $I_2$ has for its arguments in $\psi(b_1, b_2, d_{02}, a_{(12,j_{12})}, d_{12}, a_{(22,j_{22})}, \ldots, a_{(r2,j_{r2})}, d_{r2})$ but (w.l.o.g) evaluate to True and False respectively when interpreted in $M_2$.

We now show that both of the above do not occur.

Consider 1. Suppose there are 2 different instances $I_1$ and $I_2$ of a predicate $P$ which have the same set of values to their arguments in $\psi(b_1, b_2, d_0, a_{(1,j_1)}, d_1, a_{(2,j_2)}, \ldots, a_{(r,j_r)}, d_r)$ but the former is True and latter False when interpreted in $M_2$. Since $\varphi \in QPS_\Sigma(\sigma)$, if $P \in \sigma$, then since all its arguments in $V(\varphi(x)) \cup AV(\varphi(x))$, as seen earlier, it is interpreted in $M_2$ exactly as in $M$ restricted to $A_2$. Then this conflict must have been present in $M$ also, which is not the case. If $P \in \Sigma \setminus \sigma$, then since $\varphi \in QPS_\Sigma(\sigma)$, we have the following cases:

1. All arguments of both $I_1$ and $I_2$ are from $V(\varphi(x)) \cup AV(\varphi(x))$. Then as seen earlier, their interpretations in $M_2$ are exactly as in $M$ restricted to $A_2$. Then this conflict must have been present in $M$ too, which is not the case.

2. $I_1$ and $I_2$ have different argument lists. Then there is a variable $v_s$ from $EV(\varphi(x))$ which appears as the $i^{th}$ argument of $I_1$ but not as the $i^{th}$ argument of $I_2$. Now the $i^{th}$ argument of $I_2$ is a variable $v \in V(\varphi(x)) \cup EV(\varphi(x))$. Then the value substituted for $v_s$ is $a_{s,u}$ and for $v$ is either some $b$ value or some $a_{t,w}$ ($t \neq s$). Either of the latter 2 values of $v$ are different from $a_{s,u}$. Clearly $I_1$ and $I_2$ then do not have the same set of values to their arguments in which is a contradiction.

3. $I_1$ and $I_2$ have the same argument lists. Since $I_1$ and $I_2$, for the values to their arguments in $\psi(b_1, b_2, d_0, a_{(1,j_1)}, d_1, a_{(2,j_2)}, \ldots, a_{(r,j_r)}, d_r)$ were True and False when interpreted in $M_2$, they were there were True and False respectively when interpreted in $M$, for the values of their arguments in $\psi(b_1, b_2, d_0, e_1, d_1, e_2, \ldots, e_r, d_r)$. (the interpretations in $M_2$ of $I_1$ and $I_2$ were defined this way.) But since $I_1$ and $I_2$ have the same argument lists, they had the same values to their variables in $\psi(b_1, b_2, d_0, e_1, d_1, e_2, \ldots, e_r, d_r)$. This means there is a conflict in $M$.

Thus there are no internal conflicts.

Consider 2. Suppose there are 2 instances $I_1$ and $I_2$ of $P$ such that $I_1$ in

$\phi_1 = \psi(b_1, b_2, d_{01}, a_{(11,j_{11})}, d_{11}, a_{(21,j_{21})}, \ldots, a_{(r1,j_{r1})}, d_{r1})$ has the same set of values to its arguments as $I_2$ in

$\phi_2 = \psi(b_1, b_2, d_{02}, a_{(12,j_{12})}, d_{12}, a_{(22,j_{22})}, \ldots, a_{(r2,j_{r2})}, d_{r2})$ but these evaluate to different
truth values when interpreted over $M_2$. Clearly if all the arguments of both instances were in $V(\varphi(x)) \cup AV(\varphi(x))$, then as shown above, $M$ would contain conflicts. (Hence $P$ cannot be in $\sigma$.) Hence consider the case when at least one argument of at least one of $I_1$ or $I_2$ is in $EV(\varphi(x))$.

Let the instances of $P$ in $\psi$ be numbered from 1 to $q$ from left to right. Let $I_1$ be the $m^{th}$ and $I_2$ be the $n^{th}$ instance respectively.

Suppose $m = n$. Then $I_1$ and $I_2$ have the same arguments (and not just the same set of values to their arguments). Consider the largest index $j$ such that $v_j \in EV(\varphi(x))$ is an argument of $I_1$ (and hence of $I_2$). Since $\varphi \in \text{QPS}_\sigma(\Sigma)$, $v_j$ is the highest indexed variable of $I_1$ and $I_2$. Now let the common value of $v_j$ in $\phi_1$ and $\phi_2$ be $a_{j,u}$. Then while we were constructing the interpretations in $M_2$ of all the predicates by moving left to right in the quantifier prefix sequence of $\varphi$, as soon as the value for $v_j$ is substituted, the values for all the arguments of $I_1$ (and $I_2$) would have been specified (since $v_j$ is the highest indexed variable of $I_1$ and $I_2$). Then in constructing $\phi_1$ and $\phi_2$, the configuration of the states of all predicate instances upon substitution of $a_{j,u}$ for $v_j$ is $u$. Since $u$ is a configuration, it defines exactly one state for all predicate instances and in particular for the $m^{th}$ instance, it defines exactly one truth value. Then, both $I_1$ and $I_2$ must have the same truth value for the values of their arguments in $\phi_1$ and $\phi_2$ when interpreted in $M_2$. But this is a contradiction.

Suppose $m \neq n$. Then since $\varphi \in \text{QPS}_\sigma(\Sigma)$ we have the following cases:

1. There is a $v_j \in EV(\varphi(x))$ which (w.l.o.g) appears as the $i^{th}$ argument of $I_1$ while the $i^{th}$ argument $v$ of $I_2$ is in $V(\varphi(x)) \cup EV(\varphi(x))$. Then the value substituted for $v_j$ is $a_{j,u}$ and for $v$ is either some $b$ value or some $a_{t,w}(t \neq j)$. Either of the latter 2 values of $v$ are different from $a_{j,u}$. Clearly $I_1$ and $I_2$ then do not have the same set of values to their arguments - a contradiction.

2. $I_1$ and $I_2$ have the same argument lists. Then consider the instance $I_1$ appearing in $\phi_2$. Since $I_1$ and $I_2$ have the same argument lists, they have the same values to their arguments in $\phi_2$. But $I_1$ in $\phi_1$ and $I_2$ in $\phi_2$ have the same values to their arguments. So $I_1$ in $\phi_1$ and $I_1$ in $\phi_2$ have the same argument lists. Now suppose (w.l.o.g) $I_1$ in $\phi_1$ is True while $I_2$ in $\phi_2$ is False when interpreted in $M_2$. Then since there are no internal conflicts, $I_1$ in $\phi_2$ is also False when interpreted in $M_2$. Thus $I_1$ in $\phi_1$ conflicts with $I_1$ in $\phi_2$ - which, as shown above, is not possible.

Thus there are no external conflicts as well.

Thus the interpretations of the predicates of $\Sigma \setminus \sigma$ in $M_2$ are conflict-free. Note that since the $\sigma$ predicates have only arguments of kind (A) described above, they are defined in $M_2$ exactly as they are defined in $M$ restricted to $A_2$ (and hence are also conflict-free). Thus if $M'_2$ is the sub-structure of $M$ generated by $A_2$, then $M'_2|\sigma = M_2|\sigma$.

Now note that by our construction, all the predicate instances, when interpreted in $M_2$ for the values of their arguments in $\psi(b_1, b_2, d_0, a_{1,j_1}, d_1, a_{2,j_2}, \ldots, a_{r,j_r}, d_r)$ have the
same states (i.e. truth values) as they have when interpreted in $M$, for the values of their arguments in $\psi(b_1, b_2, d_0, e_1, d_1, e_2, \ldots, e_r, d_r)$. Since $M \models \varphi(b_1)$, the latter evaluates to True and hence the former also evaluates to True. In other words, $M_2 \models \varphi(b_1)$.

From the preceding two paragraphs, $\varphi(x) \in \text{EBS}_{\Sigma}(\sigma)$.

**Closure Properties of QPS**

The closure properties of QPS are the same as given in the paper. Only their proofs are slightly modified due to the modified definition of the QPS property.

Let $\varphi_i(x_i) \in \text{QPS}$. Then $\varphi_i(x_i) \in \text{QPS}_{\Sigma_i}(\sigma_i)$ for some $\sigma_i \in \Sigma_i$ where $\Sigma_i = \text{Sig}(\varphi_i)$. Let $\mathcal{C} = (\Sigma_1 \cap \Sigma_2 = \sigma_1 \cap \sigma_2)$. Let $\Sigma = \Sigma_1 \cup \Sigma_2$ and $\sigma = \sigma_1 \cup \sigma_2$.

1. Closure under $\oplus$ ($\oplus = \wedge, \lor$)

   Let $\varphi(x) = \varphi_1(x_1) \oplus \varphi_2(x_2)$. Firstly assume that there are no primed variables in either $\varphi_1$ or $\varphi_2$. Then rename each quantified variable $v$ of $\varphi_2$ to $v'$. Let this new formula be $\varphi'_2(x_2)$. Now bring $\varphi$ in prenex normal form with the leftmost $\exists$ quantified variables of $\varphi$ being the leftmost $\exists$ quantified variables of $\varphi_1$ and $\varphi'_2$.

   Then $V(\varphi(x)) = V(\varphi_1(x_1)) \cup V(\varphi'_2(x_2))$, $AV(\varphi(x)) = AV(\varphi_1(x_1)) \cup AV(\varphi'_2(x_2))$, $EV(\varphi(x)) = EV(\varphi_1(x_1)) \cup EV(\varphi'_2(x_2))$.

   Consider $P \in \sigma$ of arity $\geq 1$. Then we have the following cases:

   (a) $P \in \sigma_1 \cap \sigma_2$. Then, an instance of $P$ in $\varphi$ is either an instance of $P$ in $\varphi_1$ or $\varphi'_2$. In the former case, all its arguments are in $V(\varphi_1(x_1)) \cup AV(\varphi_1(x_1))$ and hence in $V(\varphi(x)) \cup AV(\varphi(x))$. In the latter case, all its arguments are in $V(\varphi'_2(x_2)) \cup AV(\varphi'_2(x_2))$ and hence in $V(\varphi(x)) \cup AV(\varphi(x))$.

   (b) $P \in \sigma_1 \setminus \sigma_2$ or $P \in \sigma_2 \setminus \sigma_1$. Now since $\mathcal{C}$ holds, $(\sigma_1 \setminus \sigma_2) \cap \Sigma_2 = \emptyset$ and $(\sigma_2 \setminus \sigma_1) \cap \Sigma_1 = \emptyset$. Then for $P \in \sigma_1 \setminus \sigma_2$, every instance of $P$ in $\varphi$ is an instance of $P$ in $\varphi_1$. Then all its arguments are in $V(\varphi_1(x_1)) \cup AV(\varphi_1(x_1))$ and hence in $V(\varphi(x)) \cup AV(\varphi(x))$. Similarly, for $P \in \sigma_2 \setminus \sigma_1$, for any instance of $P$, all its arguments are in $V(\varphi(x)) \cup AV(\varphi(x))$.

   Consider now $P \in \Sigma \setminus \sigma$. Again since $\mathcal{C}$ holds, $(\Sigma_i \setminus \sigma) \subseteq (\Sigma_i \setminus \Sigma_j)$, $i, j \in \{1, 2\}, i \neq j$. Then consider $P \in \Sigma \setminus \sigma$. Then $P \in \Sigma_1 \setminus \sigma$ or $P \in \Sigma_2 \setminus \sigma$. Then either (a) $P \notin \Sigma_2$ and $P \in \Sigma_1 \setminus \sigma$ and hence every instance of $P$ appears only in $\varphi_1$ or (b) $P \notin \Sigma_1$ and $P \in \Sigma_2 \setminus \sigma$ and hence every instance of $P$ appears only in $\varphi_2$.

   In either case, it is easy to see that the QPS conditions are satisfied by $P$.

   Thus $\varphi(x) \in \text{QPS}_{\Sigma}(\sigma) \subseteq \text{QPS}$.

2. $\neg$ Non-closure

   QPS is also not closed under negation. Consider $\varphi = \exists x \forall y (P(x, x) \land P(y, y))$. Clearly $\varphi \in \text{QPS}$. But in $\neg \varphi = \forall x \exists y \neg (P(x, x) \land P(y, y))$, the exists quantified $y$ as the 1$^{st}$ argument of $P(y, y)$ comes after the $\forall$ quantified $x$ in the 1$^{st}$ argument of $P(x, x)$ in the prefix structure of $\neg \varphi$. Hence $\neg \varphi \notin \text{QPS}$.
3. $\exists -$ closure.
Under $\exists$ quantification, we note that the sets $V(\cdot), AV(\cdot)$ and $EV(\cdot)$ remain unchanged and the highest indexed argument of any predicate instance continues to remain its highest indexed argument. Hence $QPS$ is closed under $\exists$ quantification.

4. $\forall -$ Non-closure
Consider the formula $\varphi(x) = \exists y(P(y) \lor P(x))$. Clearly $\varphi \in QPS_{\{P\}}(\emptyset)$. But $\forall x \varphi(x) \notin QPS_{\Sigma}(\sigma)$ for any $\sigma \subseteq \Sigma$.

Consider the sub-class of $QPS$ in which each formula $\varphi(x)$ is such that if for the variable $z$ being $\forall$ quantified, $z$ occurs as the $i^{th}$ argument of a predicate $P$, then the $i^{th}$ argument in each instance of $P$ is in $V(\varphi(x)) \cup AV(\varphi(x))$. Then under $\forall$ quantification of $z$, $V(\varphi(x)) \cup AV(\varphi(x))$ remains unchanged and the $i^{th}$ argument of each instance of $P$ continues to be in $V(\varphi(x)) \cup AV(\varphi(x))$. Its easy to check that the $\forall -$ quantified formula is in $QPS$. (More precisely, if $\varphi(z,x) \in QPS_{\Sigma}(\sigma)$, then $\forall z \varphi(z,x) \in QPS_{\Sigma}(\sigma)$.) Hence this sub-class is closed under $\forall$.

11.2 $\mathcal{EBS}$ and semi-linear sets

We first introduce some notation. Let $V$ be the set of all vocabularies. Let $\mathcal{EBS} = \bigcup_{\Sigma \in V} \bigcup_{\sigma \subseteq \Sigma} \mathcal{EBS}_{\Sigma}(\sigma)$. We look now at the relation between $\mathcal{EBS}$ and semi-linear sets. Before that, we need the following result.

Claim 6:
A sentence $\varphi \in \mathcal{EBS}$ iff either $\varphi$ is unsatisfiable or there exists $B'$ such that $\varphi$ has a model of every cardinality ranging from $B'$ to the cardinality of the $\varphi$'s largest model.

Proof:
A formula $\psi \in \mathcal{EBS}_{\Sigma}(\emptyset)$ iff $\Sigma = Sig(\psi)$ and there exists $B$ such that for every $\Sigma -$structure $M$ having universe $A$ with $|A| \geq B$ satisfying $M \models \psi$, there exists $A_1 \subseteq A$ such that

1. $|A_1| \leq B$

2. For all $A_2$, $A_1 \subseteq A_2 \subseteq A$, there exists a $\sigma -$structure $M_2$ with universe $A_2 \cup C$ such that $M_2 \models \psi$. (where $C = \{c^M|c constant , c \in \sigma\}$ is the set of interpretations in $M$ of the constants of $\sigma$).

Assume that $\varphi \in \mathcal{EBS}$. Then $\varphi \in \mathcal{EBS}_{\Sigma}(\emptyset)$ where $\Sigma = Sig(\varphi)$. Then either $\varphi$ is unsatisfiable or there is a $B$ such that for the largest model $M$ of $\varphi$, the size of its universe $A$ is $\geq B$ and there is $A_1 \subseteq A$ satisfying the properties as mentioned above. Then take $B' = B + |C|$. Then for every $A_2$ such that $A_1 \cup C \subseteq A_2 \subseteq A$, there exists a $\Sigma -$structure $M_2$ with universe $A_2$ satisfying $M_2 \models \varphi$. Then the 'only-if' part of the Lemma holds.
We now prove the 'if' part. Assume that the 'if' part is true for a formula \( \varphi \). Let \( \Sigma = \text{Sig}(\varphi) \) and consider a \( \Sigma \)-structure \( M \) having universe \( A \) with \( |A| \geq |B| \) where \( B = B' + |C| \) such that \( M \models \varphi \). Consider any subset \( A_1 \) of \( A \) such that \( |A_1| = |B| \). Consider \( A_2 \) such that \( A_1 \subseteq A_2 \subseteq A \). Now since \( C \subseteq A \), we have \( A_1 \subseteq A_2 \cup C \subseteq A \). Then \( B \leq |A_2 \cup C| \leq \text{size of } \varphi \text{'s largest model} \). Then by the 'if' part, there is a model \( M'_2 \) of \( \varphi \) of cardinality \( |A_2 \cup C| \). Then consider the model \( M_2 \) with universe \( A_2 \cup C \) and isomorphic to \( M'_2 \). Since boolean queries are closed under isomorphism [LL98], \( M_2 \models \varphi \). Thus \( \varphi \in \text{EBS}_\Sigma(\emptyset) \) and hence \( \varphi \in \text{EBS} \).

We now look at the relation between \( \text{EBS} \) and semi-linear sets.

By Claim 6, if \( \varphi \in \text{EBS} \), either \( \varphi \) is unsatisfiable or there is a \( B' \) such that \( \varphi \) has models of all cardinalities ranging from \( B' \) to the cardinality of \( \varphi \)'s largest model. The spectrum of a formula \( \varphi \) is defined as the set of the cardinalities of all its models. Let \( S_\varphi \) be the spectrum of \( \varphi \) restricted to countable cardinalities. Then for \( \varphi \in \text{EBS} \), \( S_\varphi \) is of the form

1. \( S_\varphi = \{B_{01}, B_{02}, \ldots, B_{0n}\} \) if \( \varphi \) has no infinite model. Here \( B_{0i} < B_{0(i+1)} \).
2. \( S_\varphi = \{B_{01}, B_{02}, \ldots, B_{0n}\} \cup \{B', B' + 1, B' + 2, \ldots\} \) if \( \varphi \) has a finite model. Here \( B_{0i} < B_{0(i+1)}, B_{0n} < B' \).

A linear set produced by the pair \((p, q), p, q \in \mathbb{N}\) is defined as \( \{p + iq | i \in \mathbb{N}\} \). A semi-linear set is a finite union of linear sets. Then we see that \( S_\varphi \) is a finite union of linear sets produced by the pairs (a) \((B_{0i}, 0), 1 \leq i \leq n\) for (1) and (b) \((B_{0i}, 0), 1 \leq i \leq n\) and \((B', 1)\) for (2). Thus for \( \varphi \in \text{EBS} \), \( S_\varphi \) is a semilinear set.

What can we say about the converse? That is, given a semi-linear set \( S \), is there \( \varphi \in \text{EBS} \) such that \( S_\varphi = S \)? Clearly, if it is, then as seen above it is a semi-linear set with \( q \) being 0 or 1. But \( q \), for a general semi-linear set is not restricted to be 0 or 1. So for the converse to also be true, we must consider a class larger than \( \text{EBS} \). We then relax the definition of the EBS property as below:

A formula \( \varphi(x) \) is said to satisfy the Extensible Bounded Submodel property with respect to \( \sigma \subseteq \Sigma \) (denoted as \( \varphi(x) \in \text{EBS}_\Sigma(\sigma) \)) if \( \text{Sig}(\varphi) = \Sigma \) and there exists a cardinal \( B \) (dependent on \( \varphi \) and \( \sigma \) in general) such that for every \( \Sigma \)-structure \( M \) having universe \( A \) with \( |A| \geq B \) and for every \( a \in A^{|x|} \), if \( M \models \varphi(a) \) then there exists \( A_1 \subseteq A \) such that

1. \( a \in A_1^{|x|} \)
2. \( |A_1| \leq B \)
3. There exists a recursive function \( f_\varphi : \mathbb{N} \rightarrow S_\varphi \) with the property that \( f_\varphi(x) \) is the smallest number in \( S_\varphi \) greater than \( x \). Then
11.2. \( \mathcal{EB}_S \) AND SEMI-LINEAR SETS

(a) for all \( A_2 \) such that \( A_1 \subseteq A_2 \subseteq A \) and \( |A_2| < \aleph \) there exists \( A'_2 \) such that \( A_2 \subseteq A'_2 \subseteq A \), \( |A'_2 \cup C| = f_\varphi(|A_2|) \) (\( C \) is the set of interpretations of the constants of \( \Sigma \)) and if \( M'_2 \) is the substructure of \( M \) generated by \( A'_2 \), then there exists a \( \Sigma \)-structure \( M_2 \) such that \( M_2|_\sigma = M'_2|_\sigma \) and \( M_2 \models \varphi(a) \).

(b) for all \( A_2 \) such that \( A_1 \subseteq A_2 \subseteq A \) and cardinality of \( A_2 \) is infinite, if \( M'_2 \) is the substructure of \( M \) generated by \( A_2 \), then there exists a \( \Sigma \)-structure \( M_2 \) such that \( M_2|_\sigma = M'_2|_\sigma \) and \( M_2 \models \varphi(a) \).

Note the following:

- By the above definition, \( f_\varphi \) is a monotone function.
- The size of \( S_\varphi \) must be infinite. Suppose it was finite, it would have a largest element say \( t \). Then \( f_\varphi(t) \) will not be defined.
- If the largest model of \( \varphi \) has finite size, then there cannot exist any \( f_\varphi \) satisfying the above conditions. Suppose it existed. Let \( A \) be the universe of the model. Then choosing \( A_2 = A \), there must exist \( A'_2 \subseteq A \) such that \( |A'_2 \cup C| = f_\varphi(|A|) > |A| \) and satisfying the conditions of 3a. But since \( A'_2 \cup C \subseteq A \), we have a contradiction.

Now, since the motivation behind this relaxation is to achieve the equivalence that a sentence \( \varphi \in \mathcal{EB}_S \) iff \( S_\varphi \) is semi-linear, we might have to impose some additional constraints on \( f_\varphi \). We now investigate what these additional constraints are.

We firstly have the following.

Claim 7:
If a sentence \( \varphi \in \mathcal{EB}_S \), then

1. \( \varphi \) is unsatisfiable or the largest model of \( \varphi \) is of finite size or

2. For some \( n \geq 0 \), \( S_\varphi = \{B_{01}, B_{02}, \ldots, B_{0n}\} \cup \{f^{(i)}_\varphi(B) | i \geq 1\} \).

Proof:
(1) is easy to see. For (2), since \( \varphi \in \mathcal{EB}_S \), \( \varphi \in \mathcal{EB}_{S_{\Sigma}}(\sigma) \). Then clearly there exist models of sizes \( f^{(i)}_\varphi(B), i \geq 1 \). Further by the property that \( f_\varphi(x) \) is the smallest number in \( S_\varphi \) greater than \( x \), we see that there are no models of sizes in between \( f^{(i)}_\varphi(B) \) and \( f^{(i+1)}_\varphi(B) \) for all \( i \geq 1 \).

We recall that the modified definition of the \( \mathcal{EB}_S \) property was to achieve the equivalence that a formula \( \varphi \in \mathcal{EB}_S \) iff \( S_\varphi \) is semi-linear. Then we need to know what are functions \( f_\varphi \) such that \( S_\varphi \) is semi-linear.

A property of semi-linear sets
We state the following without proof.
Claim 8:
Consider a set \( S \subseteq \mathbb{N} \). Consider the sequence \( s \) of elements of \( S \) ordered by \(<\). Then consider the ‘difference’ sequence \( d \) of \( S \) which is formed by taking the difference of consecutive elements of \( s \). (Eg. if \( s = \{1, 3, 7, 14, \ldots \} \), \( d = 2, 4, 7, \ldots \)).

Then \( S \) is semi-linear iff \( d \) is of the form

\[
a_1, a_2, a_3, \ldots, a_n(b_1, b_2, \ldots, b_m)^* \]

(* means repetition infinite number of times)

Then if \( S_\varphi \) is semi-linear we have the following:

1. Either it is finite or
2. For some \( n_0 \geq 0 \),

\[
f_\varphi^{(i)}(B) - f_\varphi^{(i-1)}(B) = b_j \quad i = n_0 + j + km, 1 \leq j \leq m, k = 0, 1, \ldots
\]

Thus for \( i > n_0 \), the difference \( D(i) = f_\varphi^{(i)}(B) - f_\varphi^{(i-1)}(B) \) is a periodic function.

(Note that if \( S_\varphi \) was semi-linear with the generating pairs of the form \( (p, 0) \) and atleast one pair \( (p, 1) \), then the difference sequence of \( D \) is of the form \( a_1, a_2, \ldots, a_n(1)^* \). Then \( D(i) = f_\varphi^{(i)}(B) - f_\varphi^{(i-1)}(B) = 1 \) for \( i > n_0 \). Then \( f_\varphi(x) = x \) for all \( x \geq f_\varphi^{n_0}(B) \). We can see that this special case is indeed the old definition of EBS with the cardinal \( f_\varphi^{n_0}(B) \) being taken as the cardinal satisfying the EBS conditions.)

We now add this condition to the relaxed EBS definition and state the modified definition below.

A formula \( \varphi(x) \) is said to satisfy the Extensible Bounded Submodel property with respect to \( \sigma \subseteq \Sigma \) (denoted as \( \varphi(x) \in \text{EBS}_\Sigma(\sigma) \)) if \( \text{Sig}(\varphi) = \Sigma \) and there exists a cardinal \( B \) (dependent on \( \varphi \) and \( \sigma \) in general) such that for every \( \Sigma \)-structure \( M \) having universe \( A \) with \( |A| \geq B \) and for every \( a \in A^{[x]} \), if \( M \models \varphi(a) \), then there exists \( A_1 \subseteq A \) such that

1. \( a \in A_1^{[x]} \)
2. \( |A_1| \leq B \)
3. There exists a recursive function \( f_\varphi : \mathbb{N} \rightarrow S_\varphi \) with the property that (a) \( f_\varphi(x) \) is the smallest number in \( S_\varphi \) greater than \( x \) (b) there exists some \( n_0 \) such that for \( i > n_0 \), the difference \( D(i) = f_\varphi^{(i)}(B) - f_\varphi^{(i-1)}(B) \) is a periodic function. Then

(a) for all \( A_2 \) such that \( A_1 \subseteq A_2 \subseteq A \) and \( |A_2| < \mathbb{N} \) there exists \( A'_2 \) such that \( A_2 \subseteq A'_2 \subseteq A \), \( |A'_2 \cup C| = f_\varphi(|A_2|) \) (\( C \) is the set of interpretations of the constants of \( \Sigma \)) and if \( M'_2 \) is the substructure of \( M \) generated by \( A'_2 \), then there exists a \( \sigma \)-structure \( M_2 \) such that \( M_2|_\sigma = M'_2|_\sigma \) and \( M_2 \models \varphi(a) \).
(b) for all $A_2$ such that $A_1 \subseteq A_2 \subseteq A$ and cardinality of $A_2$ is infinite, if $M'_2$ is the substructure of $M$ generated by $A_2$, then there exists a $\sigma$–structure $M_2$ such that $M_2|_\sigma = M'_2|_\sigma$ and $M_2 \models \varphi(a)$.

As before, let $\mathcal{EBS} = \bigcup_{\Sigma \subseteq V} \bigcup_{\sigma \subseteq \Sigma} \mathcal{EBS}_\Sigma(\sigma)$. Again as before, we have $\mathcal{EBS}_\Sigma(\sigma_1) \subseteq \mathcal{EBS}_\Sigma(\sigma_2)$ if $\sigma_2 \subseteq \sigma_1 \subseteq \Sigma$. Hence $\mathcal{EBS} = \bigcup_{\Sigma \subseteq V} \mathcal{EBS}_\Sigma(\emptyset)$.

We now prove that $\varphi \in \mathcal{EBS}$ iff $S_\varphi$ is semi-linear.

Proof:

Let $S_\varphi$ be semi-linear, then (a) either it is finite or (b) its difference sequence is of the form $a_1,\ldots,a_n(b_1,\ldots,b_m)^*$. If it is finite, then take $B$ to be one more than the size of the largest model of $\varphi$. Clearly then $\varphi \in \mathcal{EBS}$. Else consider the function $f_\varphi : \mathbb{N} \to S_\varphi$ defined as below. Let $B_0$ be the size of the smallest model of $\varphi$, $X(j) = \Sigma_{i=1}^n a_i + B_0$, $C_1 = X(n)$, $Y(j) = \Sigma_{i=1}^n b_i$, $C_2 = Y(m)$ and $Z(k,j) = Y(j) + C_1 + k.C_2$.

$$
\begin{align*}
f_\varphi(x) &= \begin{cases} X(0) & x \leq B_0 \\
X(k) & X(k-1) < x \leq X(k), 1 \leq k \leq n \\
Z(k,j) & Z(k,j-1) < x \leq Z(k,j), 1 \leq j \leq m, k \geq 0
\end{cases}
\end{align*}
$$

Let $B = |x|$ (No. of free vars of $\varphi(x)$.) Its easy to see that $f_\varphi$ is recursive and satisfies the following:

1. $f_\varphi(x)$ is the smallest element of $S_\varphi$ which is greater than or equal to $x$.

2. For some $r \geq 0$, $S_\varphi = \{B_0,\ldots,B_{0r}\} \cup \{f_\varphi^{(i)}(B) | i \geq 1\}$.

3. Let $n_0$ be the smallest $i-1$ such that $f_\varphi^{(i)}(B) - f_\varphi^{(i-1)}(B) = b_1$. Then

$$
f_\varphi^{(i)}(B) - f_\varphi^{(i-1)}(B) = b_j \quad i = n_0 + j + km, 1 \leq j \leq m, k = 0,1,\ldots
$$

Thus for $i > n_0$, the difference $D(i) = f_\varphi^{(i)}(B) - f_\varphi^{(i-1)}(B)$ is a periodic function.

Then following the proof of Claim 6, its easy to show that $\varphi \in \mathcal{EBS}_\Sigma(\emptyset) \subseteq \mathcal{EBS}$.

Conversely assume $\varphi \in \mathcal{EBS}$. Then $\varphi \in \mathcal{EBS}_\Sigma(\sigma)$ for some $\sigma \in \Sigma$. Then for some $r \geq 0$, $S_\varphi = \{B_0,\ldots,B_{0r}\} \cup \{f_\varphi^{(i)}(B) | i \geq 1\}$. Now since for some $n_0 \geq 0$, for all $i > n_0$, $D(i) = f_\varphi^{(i)}(B) - f_\varphi^{(i-1)}(B)$ is a periodic function, we have that the difference sequence of $S_\varphi$ is of the form $a_1,\ldots,a_n(b_1,\ldots,b_m)^*$. Then $S_\varphi$ is semi-linear.

Of course with the last definition of the $\textsf{EBS}$ property, the closure properties are not guaranteed!
Chapter 12

Conclusion and Future Work

We looked at graph transformation systems in which each state is a typed graph and transitions from one state to another involve graph transformations. We are then interested in knowing whether certain safety properties are true in all the reachable states of the system given a set of initial states. This can be translated into the problem of reachability of states violating the safety property from the initial set of states. We illustrated this with the particular example of the library system which can be modeled as a graph transforming system. We then introduced the safety property of checking whether there exists a book on shelf yet there is a pending claim for the title of the book.

We then looked at how the problem of reachability analysis is undecidable in even infinite state graph transforming systems. (The problem is known to be undecidable in infinite state systems in general). We then developed a procedure InductionSolve which uses a combination of inductive reasoning and bounded model checking to perform reachability analysis in infinite state systems in general. (The procedure though is not guaranteed to terminate on all inputs.) We then gave a logical formalism for graph transforming systems where a graph transforming system (GTS) is expressed using a set of operations and a set of sub-actions on nodes and edges. We considered a special kind of sub-actions and posed the question that whether the transition relation $T(s, s')$ on states defined by an arbitrary GTS can also be defined by a GTS using the special sub-actions. The question though still open, we showed that for GTSes satisfying a certain condition, it is indeed possible to construct a GTS using the special sub-actions which defines the same transition relation on states.

We looked at the procedure InductionSolve which uses a validity checker and found that there are are two ways of non-termination of the procedure - one by an unbounded increase in its iterator, which cannot be avoided in general and the other by the non-termination of its validity checker. We saw that the bounded model property for the formulae of InductionSolve enables us to decide their validity and thus enables us to construct a terminating validity checker. We looked at 2 classes of GTSes which have bounded models for $T(s, s')$ and showed that the library system is indeed in one of these
classes. We finally considered certain safety properties and initial conditions which have the bounded model property and then showed how this can be used to get bounded models for the formulae of InductionSolve and computed expressions for the bounds. We used this to construct a decision procedure (validity checker) for the formulae of InductionSolve. Finally we showed that in the library system, for the particular safety property mentioned above, we can construct a decision procedure for it and find that InductionSolve too terminates for it and proves that the library system indeed satisfies the safety property.

Some of the future work is mentioned below:

1. We firstly note that the results in chapter 9 talk about GTSes satisfying certain semantic conditions and mention one example for each, each example satisfying certain syntactic properties which ensure the semantic conditions. We would firstly like to replace the semantic conditions by syntactic conditions while still preserving the generality of the semantic conditions. It would be even better to get a syntactic characterization which is more general than the semantic conditions. However, if we are unable to do either of these, we would atleast like to get more examples satisfying syntactic properties which ensure the semantic conditions too. It would be even better if we could, just as it was done for the classical decision problem (of checking whether an arbitrary FO formula is satisfiable), classify all the formulae on syntactic criteria into 2 sets of classes - one set containing classes for which we can decide whether formulae of the class satisfy the semantic conditions and the other set containing classes for which the problem of checking whether a formula of the class satisfies the semantic conditions is undecidable.

2. We would like to know how expressive the formalism for graph transformation systems introduced in Chapter 5 is.

3. We would like to know whether the transition relation captured by an arbitrary GTS can be captured by a GTS using the special sub-actions mentioned in Chapter 6.

4. The concept of Ehrenfeucht-Fraïssé games is very relevant to our problem. We briefly this concept is as follows:

The classical Ehrenfeucht-Fraïssé game on two \(\sigma\)–structures \(A\) and \(B\) consists of two players - a spoiler and a duplicator. The game is played in rounds and in each round there are two moves - one by the spoiler and the other by the duplicator. The spoiler selects one of the structures and chooses an element from it. The duplicator then chooses an element from the other structure. The objective of the duplicator is to show that \(A\) and \(B\) are isomorphic while that of the spoiler is to show they are not. More precisely, in a \(k\) round Ehrenfeucht-Fraïssé game, the duplicator tries to show a partial isomorphism between \(A\) and \(B\). This is the winning condition for the duplicator. If there is a strategy for the duplicator to show a partial isomorphism regardless of the spoiler’s moves, it is denoted as \(A \equiv_k B\). The theorem by Ehrenfeucht and
Fraïssé says that $\mathcal{A} \equiv_k \mathcal{B}$ iff $\mathcal{A}$ and $\mathcal{B}$ agree on all FO sentences of quantifier rank at most $k$.

We then wish to identify classes of $\sigma_S$—structures (i.e. states as defined in Chapter 5) such that each structure is $\equiv_k$ to some structure of bounded size. We then identify the properties of these classes of structures which are expressible in FO. Then we are guaranteed by the Ehrenfeucht and Fraïssé theorem that the properties are true in structures of these classes if they are true in structures of bounded size. In other words, these properties will have bounded models. What are such classes of structures and what are the properties is something we wish to explore.

5. While bounded model property is one way of deciding the formulae $IS_1(k)$ and $IS_2(k)$, we would like to explore other ways of deciding these formulae.
References


