International Journal of Psychosocial Rehabilitation, Vol. 24, Issue 04, 2020
ISSN: 1475-7192

Ordered Binary Decision Diagram-based Decision algorithm for Iteration-free CPDL

[1]Leelavathi Rajamanicham, [2]Qin FenPing, [3]Chen Jia

Abstract: *Propositional dynamic logic is one of the simplest applied modal logics designed for reasoning about the behavior of programs. Iteration-free converse propositional dynamic logic is an iteration-free fragment of propositional dynamic logic with converse of programs. Starting from a converse propositional dynamic logic formulas, the algorithm introduces the negative converse normal form transformation rule and the FLAT rule to do some preprocessing; then the model of set of formulas is reconstructed and transformed into some Boolean formulas; finally, these Boolean formulas are represented as ordered binary decision diagram, based on the existing ordered binary decision diagram software package that can be called for deciding the satisfiability of set of formulas. Proves that the algorithm is terminating, sound and complete, then realize the decision algorithm of iteration-free converse propositional dynamic logic.*

Keyword: *propositional dynamic logic; satisfiability-checking; ordered binary decision diagram*

I. INTRODUCTION

Propositional Dynamic Logic (PDL) was originally introduced by Fischer and Ladner in 1979. It is used for formal description and reasoning of programs and can provide an appropriate framework for modeling and reasoning actions. One of the basic reasoning problems in propositional dynamic logic is to check the satisfiability of the formula set, and other reasoning problems can usually be reduced to this problem. Fischer and Ladner[1] prove that the satisfiability of Propositional Dynamic Logic（PDL） is exponentially complete. With the extensive application of propositional dynamic logic in the fields of program verification, action theory and knowledge representation, it is of great significance to study the satisfiability determination algorithm of propositional dynamic logic.In general, the effective decision process of modal logic and description logic is based on Tableau algorithm[2]. Pratt[3] proposed the original

1 School of Information Technology, SEGi University, Malaysia.

2 School of Information Technology, SEGi University, Malaysia.

3 School of Information Technology, SEGi University, Malaysia.

Received: 03 Dec 2019 | Revised: 21 Jan 2020 | Accepted: 18 Feb 2020

Tableau decision algorithm for propositional dynamic logic. Its essence is to construct an and-or graph for the formula set under consideration through table rules and global buffering, so as to check whether the model of the formula set can be extracted from the graph. Tableau algorithm is the most important reasoning algorithm in propositional dynamic logic at present, but it does not perform best in all cases, so it is necessary to find a more appropriate reasoning algorithm.

Ordered binary decision diagram (OBDD) is a data structure that can effectively represent and deal with large-scale problems. It has been successfully applied in the field of large-scale model detection and verification, and also has great application potential in determining the satisfiability of logic formulas. Binary decision diagram was successfully applied to proposition logic and model checking, and then promoted to the level of first-order logic. Pan [4] applies binary decision diagram-based description forms to modal logic K, and uses binary decision diagram to represent and use various types of sets to study the basic representation schemes of different types of sets. The results show that this binary decision diagram - based method has an advantage in formula reasoning for modal heavy forms. The corresponding relationship between propositional dynamic logic and description logic was first proposed in Schild's article [5], which is based on the mapping relationship between description logic knowledge base model and proposition dynamic logic special formula model. On the extension level, description logic's individuals (members in ***I***) correspond to propositional dynamic logic 's state, while the connection between the two individuals corresponds to the transition between the two States. On the connotation level, concepts correspond to propositions and roles correspond to atomic actions. More precisely, Schild pointed out that converse propositional dynamic logic corresponds to description logic ALCIreg. description logic ALCIreg is in the role by removing the full name value restriction on ALCQI and adding the regular expression of the constructor. Rudolph et al [6]applied ordered binary decision diagram to the reasoning of the description logic language SHIQ terms, pointing out that the SHIQ knowledge base can meet the equivalence reduction into the ALCIb knowledge base, and establishing model theoretical results for the ALCIb, putting forward the decision process and proving its completeness and reliability. Bambini [7] gives the axiom system of Iteration-free converse propositional dynamic logic

Based on the above theory, this paper presents an algorithm for determining the satisfiability of Iteration-free converse propositional dynamic logic based on ordered binary decision diagram, which proves theoretically the reliability, completeness and termination of the algorithm. In the first section, the related theories of Iteration-free converse propositional dynamic logic and ordered binary decision diagram are described first. In the second section, the Iteration-free converse propositional dynamic logic judgment algorithm based on ordered binary decision diagram is given. Each formula is converted into an equivalent formula by using the negative converse normal form (NCNF) and the complex formula set is converted into a simple formula set by using FLAT rules. Then the model is reconstructed, and the constructed model is converted into a Boolean function, and the Boolean function is judged to be satisfiable by ordered binary decision diagram. In the third section, based on Jived software package, an inference engine adapted to the satisfiability determination of Iteration-free converse propositional dynamic logic formula set is developed, and the algorithm of Iteration-free converse propositional dynamic logic satisfiability determination is verified by an example.

II. BASIC CONCEPTS

A. Iteration-free CPDL

From a syntax point of view, Iteration-free converse propositional dynamic logic consists of two types of expressions: actions and formulas. Actions and formulas are established from the set of atomic programs, the set of propositional letters and the application of appropriate operators. In this paper,$\Pi_0$is used to represent the set of atomic actions,Φ_0 is used to represent the set of atomic propositions (such as atomic formula), σ is used to declare the element in α, p and q are used to declare the element in α, β is used to declare any action, andφ, ψ, ξ is used to represent any formula. This article focuses on Iteration-free converse propositional dynamic logic, and its main syntax is as follows:

$$\varphi ::= \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \rightarrow \psi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \langle\alpha\rangle\varphi \mid [\alpha]\varphi$$

$$\alpha ::= \sigma \mid \alpha;\beta \mid \alpha \cup \beta \mid \alpha^{-} \mid \varphi ?$$

The semantics of Iteration-free converse propositional dynamic logic are based on the transformation system (Kripkemodel). A Kripke model is a binary set **I**=<Δ^I, $\cdot^I$>,Δ^I is a set of States, $\cdot^I$ is an explanatory function, it maps each proposition p to a subset $\mathbf{p}^\mathbf{I}$ of $\Delta^\mathbf{I}$, and maps each action σ to binary relation $\sigma^\mathbf{I}$ in $\Delta^\mathbf{I}$. The interpretation function can be expanded to interpret complex formulas and complex actions as follows:

$\top^I = \Delta^I$, $\bot^I = \emptyset$

$(\neg\varphi)^I = \Delta^I \backslash \varphi^I$, $(\varphi \rightarrow \psi)^I = (\neg\varphi \vee \psi)^I$

$(\varphi \wedge \psi)^I = \varphi^I \cap \psi^I$, $(\varphi \vee \psi)^I = \varphi^I \cup \psi^I$

$(\langle\alpha\rangle\varphi)^I = \{x \in \Delta^I \mid \exists y\, (\alpha^I(x, y) \wedge \varphi^I(y))\}$

$([\alpha]\varphi)^I = \{x \in \Delta^I \mid \forall y\, (\alpha^I(x, y) \rightarrow \varphi^I(y))\}$

$(\alpha \cup \beta)^I = \alpha^I \cup \beta^I, (\alpha^-)^I = \{(y, x) \mid (x, y) \in \alpha^I\}$

$(\alpha;\beta)^I = \{(x,y) \mid \exists z(\alpha^I(x,z) \wedge \beta^I(z,y))\}$

$(\varphi?)^I = \{(x, x) \mid \varphi^I(x)\}$

$\omega \in \varphi^I$ is expressed by $\boldsymbol{I}, \omega \vDash \varphi$.For the set of formulas $\boldsymbol{\Phi}$ and $\varphi \in \boldsymbol{\Phi}$, $\boldsymbol{I}, \omega \vDash \boldsymbol{\Phi}$ are used to represent $\boldsymbol{I}$, $\omega \vDash \varphi$, If $\boldsymbol{I}$, $\omega \vDash \varphi$, then $\boldsymbol{I}$ satisfies φ in the state ω. Iteration-free converse propositional dynamic logic has a finite model property: if $\boldsymbol{\Phi}$ is satisfied, a Kripke model can be constructed to satisfy $\boldsymbol{\Phi}$.

B. Ordered Binary Decision Diagram

Ordered binary decision diagram (OBDD) is an effective data structure for encoding Boolean functions. Structurally, Binary decision diagram is a directed acyclic graph: two different true and false nodes are called end nodes, and nodes without input arcs are called root nodes. With the exception of the terminal node, each node marks a variable from the variable set Var and has two output arcs that mark true and false values, respectively.

Definition 1 Binary decision diagram is an Octtuple O= (N, n_{root}, n_{true}, n_{false}, *low, high*, ***Var***, λ), where

(1) n is a finite set of nodes;

(2) $n_{root} \in N$ is the root node;

(3)n_{true}， $n_{false} \in N$ is the terminal node;

(4) Low and high, respectively representing 0 branchsub-nodes and 1 - branch sub-nodes of non-terminal nodes;

(5) ***Var*** is a finite set of variables;

(6) λ is a variable assign to each non-terminal node;

Ordered binary decision diagram differs from binary decision diagram in that the order of variables appearing on any path from the root node to the leaf node in ordered binary decision diagram remains the same. In addition, two simplified rules are introduced into Ordered binary decision diagram, the restriction of the order of variables and the restriction of the simplified rules, making ordered binary decision diagram the norm for expressing Boolean functions. Each of the following binary decision diagrams is based on the variable set ***Var***= $\{x_1, \cdots, x_n\}$ represents the n-ary Boolean function ϕ： $2^{Var} \rightarrow \{true, false\}$

Definition 2 Given a binary decision diagram, $O = (N, n_{root}, n_{true}, n_{false}, low, high, Var, \lambda)$, Boolean function ϕ_o: $2^{Var} \rightarrow \{true, false\}$ is recursively defined as follows:

$\phi_o = \phi n_{root} \phi n_{true} = [true]\ \phi n_{false} = [false]$

$\phi_n = (\neg[\lambda(n)]_x \wedge \phi_{low}(_n)) \vee ([\lambda(n)]_x \wedge \phi_{high(n)})$

$n \in N \backslash \{ n_{true}, n_{false} \}$

Where, $[v]_x$ represents a characteristic boolean function.

In general, the value $\phi(V)$ of some $V \subseteq \boldsymbol{Var}$ is determined by binary decision diagram, starting from the root node: $v \in V$ is identified on each node, if $v \in V$, then the node connects 1 side, otherwise it connects 0 sides. If the terminal node is reachable, the identity returns a result.

How to effectively construct ordered binary decision diagram and determine its satisfiability has always been a hot issue for scholars. Drechsler[8] stated that whether ordered binary decision diagram can be calculated depends mainly on the selected variable order, and the satisfiability test of Boolean functions is exponential in the worst state, but can be completed in polynomial time in the best state, but in general, The satisfiability reasoning of ordered binary decision graph is NP- complete. Therefore, using ordered binary decision graph to judge the satisfiability of Boolean functions depends on the variable order of Boolean functions.

III. ORDERD BINARY DECISION DIAGRAM -BASEDDECISIONALGORITHM FOR ITERATION-FREECONVERSE PROPOSITIONAL DYNAMIC LOGIC

First, suppose that formulas and actions appear in negative converse normal form (NCNF): conjunctions → will not appear, operators¬ appear only in front of propositions, and converse actions operators - only in front of propositions. Apply the following three rules to convert the formula into an equivalent negative converse normal form.

(1) $\neg(\varphi \wedge \psi) \equiv (\neg\varphi \vee \neg\psi)$

(2) $(\alpha;\beta)^- \equiv (\beta^-;\alpha^-)$

(3) $(\alpha \cup \beta)^- \equiv (\alpha^- \cup \beta^-)$

The time cost of the conversion process is linearly related to the size of the formula set and is limited to space, so the detailed conversion process will not be described here. The iteration-free converse propositional dynamic logic formula set described below are all negative converse normal form

For any iteration-free converse propositional dynamic logic formula set ***Φ***, it can be processed through the following steps:

(1) First, the next four rules are applied to***Φ*** to eliminate the compound action operator and test operator.

$<\alpha;\beta>\varphi=<\alpha><\beta>\varphi$

$[\alpha;\beta]\varphi=[\alpha][\beta]\varphi$

$<\varphi?>\psi=\varphi\wedge\psi$

$[\varphi?]\psi=\varphi\rightarrow\psi$

(2) Selecting the outermost layer in the form of $<U>\psi$and $[U]\psi$ in***Φ***, whereψ is a non-atomic proposition; Replaceψ with a new proposition name ξ that does not exist inX and add$(\neg\xi\sqcup\psi)$to ***Φ***.

(3) Repeat (1) (2) ,until allψ in$<U>\psi$and $[U]\psi$are atomic propositions.

For any formula set, the formula set obtained after the above steps of conversion is denoted as FLAT (*Φ*), and the following theorem can be easily proved through the above conversion process.

Theorem 1*Φ* is a finite set of NCNF formulas in any original language.*Φ* is satisfiable if and only if FLAT (*Φ*) is satisfiable.

Usually, detecting satisfiability problems is actually an attempt to construct an explanatory model. To determine the satisfiability of Iteration-free converse propositional dynamic logic formula set*Φ*, it is necessary to construct a corresponding model to determine it. For example, PDL's tableau algorithm is to construct a safe, conflict-free complete tree to determine the formula satisfiability. Therefore, how to reconstruct the Iteration-free converse propositional dynamic logicformula set will be described first, which is actually the process of determining the satisfiability of the formula set, and then its reliability and completeness will be proved.

First, we need to obtain the atomic formula set of the proposition dynamic logic formula, which will be realized by the following functions:

$$\begin{cases} CL(\psi) & \text{If } \varphi=\neg\psi \\ CL(\psi)\cup CL & \text{If } \varphi=\psi\sqcap\xi \text{ or } \psi=\psi\sqcup\xi \\ (\xi) & \text{If } \varphi=<U>\psi \text{ or } \varphi=[U]\psi \\ \{\varphi\}\cup CL(\psi) & \text{otherwise} \end{cases}$$

$CL(\varphi) \coloneqq$

Definition 3 LetΦ be the finite set of negative converse normal form formulas in the original language, given an explanation I=<Δ^I, $\cdot^I$>, $\varphi \subseteq \boldsymbol{\varphi}$ is the formula set in FLAT (Φ), and the π mapping of φ is I, denoted π (I), which contains all triples **<*A, R, B*>** of ω、ω′∈Δ^I

$\mathbf{A} = \{\varphi \subseteq \boldsymbol{\varphi} \mid \omega \in \varphi^I\}$

$\mathbf{R} = \{U \in \Pi_0 \mid <\omega, \omega'> \in U^I\}$

$\mathbf{B} = \{\varphi \subseteq \boldsymbol{\varphi} \mid \omega' \in \varphi^I\}$

Definition4 Given any Iteration-free converse propositional dynamic logic finite formula setΦ and any non-empty interpretation I= <Δ^I, $\cdot^I$>. And defines φ=P(FLAT(Φ)), $\boldsymbol{\pi_D}(I)$ as the interpretation I, which is π mapping with respect to the set $\boldsymbol{D}$. First, the sets $\boldsymbol{D_0}$and $\boldsymbol{D_1}$ are constructed as follows:

(1) $\boldsymbol{D_0}$ is a set of triples **<*A, R, B*>**that satisfy the following conditions:

Bc: If each formulaφ∈FLAT(Φ) , then ⊓$\varphi$is satisfied;

Ex: For all <U>$\psi \in \boldsymbol{\varphi}$, if $\psi \in \mathbf{B}$,**R**⊢U, then <U>$\psi \in \mathbf{A}$;

Uni:For all [U] $\psi \in \boldsymbol{\varphi}$, if[U] $\psi \in \mathbf{A}$,**R**⊢U,then $\psi \in \mathbf{B}$;

(2) $\boldsymbol{D_1}$ is a set of triples **<*A, R, B*>** that satisfy the following conditions in $\boldsymbol{D_0}$:

Delex: For all<U>$\psi \in \boldsymbol{\varphi}$, if [U]$\psi \notin \mathbf{A}$ then **R**⊢U, $\psi \in \mathbf{B}$ **;**

Deluni: For all [U]$\psi \in \boldsymbol{\varphi}$, if <U>$\psi \in \mathbf{A}$ then**R**⊢U， $\psi \in \mathbf{B}$ **;**

Sym：<**B**, *conv*(**R**), **A**>∈$\boldsymbol{D_0}$;

If $\boldsymbol{D_1}$is an empty set, make $\boldsymbol{D_\Phi} = \boldsymbol{D_0}$**;** otherwise make $\boldsymbol{D_\Phi} = \boldsymbol{D_1}$,whiche is called the reconstructed $\boldsymbol{\pi}$ set of $\boldsymbol{\Phi}$.

Theorem 2 (reliability) allows $\boldsymbol{\Phi}$ to be any Iteration-free converse propositional dynamic logic finite formula set and $\boldsymbol{D_\Phi}$ to be a reconstructed $\boldsymbol{\pi}$ set of $\boldsymbol{\Phi}$, defining any non-null interpretation $\boldsymbol{I}$ = <Δ^I, $\cdot^I$>≔ $\boldsymbol{I}$ ($\boldsymbol{D_\Phi}$). If $\boldsymbol{D_\Phi}$ is not empty, then$\boldsymbol{I} \models \boldsymbol{\Phi}$.

Proof:

Bc: No matter when $\boldsymbol{D_\Phi}$ is, the universe of I is obviously not empty. If φ is an atomic proposition, it is directly obtained according to explanation I. Only φ= <U>ψ and φ=[U] ψ are considered below.

Ex: If φ= <U>ψ, according to$\boldsymbol{D_\Phi}$is not empty, there is at least one triplet <**A**, **R**,**B**>. If $\psi\in$**B**,**R**$\vdash$U, according to ***Ex***,we can know <U>$\psi\in$**A**. And because <U>$\psi\in\varphi$, according to the inductive hypothesis and triple definition, there must be two States ω, ω ′, such thatω∈(<U>$\psi)^I$， <ω, ω′>∈U^I and ω′∈ψ^I.

Uni: If φ = [*U*] ψ, according to $\boldsymbol{D}_\Phi$ is not empty, there is at least one triple<**A**, **R**, **B**>.If [U] $\psi\in$**A**,**R**$\vdash$U , according to ***Uni*** , we can know $\psi\in$**B**. Therefore, according to the inductive hypothesis and the triple definition, for all States ω∈Δ^I, and ω∈([U]$\psi)^I$, there is another state ω′∈Δ^I such that <ω, ω′>∈U^I， ω′∈ψ^I.

Delex:Ifφ= <U> ψ, according to $\boldsymbol{D}_\Phi$ is not empty, there is at least one triple<**A**, **R**, **B**>. If <U>$\psi\in$**A**, according to the definition of triplet, there must be a state ω∈(<U>$\psi)^I$, and then according to ***Delex***, there must be U∈**R**′, $\psi\in$**B**′ forming a triplet<**A**, **R**′, **B**′>. Therefore, according to the hypothesis, there must be another state ω′∈Δ^I such that <ω, ω′>∈U^I， ω′∈ψ^I.

Deluni: If φ=[U]ψ, assuming that ω∉([U] $\psi)^I$ exists for all States ω∈Δ^I, according to the definition of triplets, [U]$\psi\notin$**A**, then the condition of ***Deluni*** is satisfied, and U∈**R**, $\psi\notin$**B,** that is <ω, ω′>∈U^I, ω′∉ψ^Ican be obtained. Therefore, there must be no triples<**A**, **R**, **B**>, which is contradictory to $\boldsymbol{D_\Phi}$.

Theorem 3 (Completeness) Allows $\boldsymbol{\Phi}$ to be any converse propositional dynamic logic finite formula set and $\boldsymbol{D_\Phi}$ to be a reconstructed $\boldsymbol{\pi}$ set of $\boldsymbol{\Phi}$, defining any non-null interpretation $\boldsymbol{I}$ = <Δ^I, $\cdot^I$>:= $\boldsymbol{I}$ ($\boldsymbol{D_\Phi}$). If $\boldsymbol{I}\models\boldsymbol{\Phi}$, then$\boldsymbol{D_\Phi}$ is not empty.

Proof:

Bc: Clearly satisfied.

Ex: according to[U] $\psi\in\boldsymbol{\varphi}$**,** $\boldsymbol{I}\models\boldsymbol{\Phi}$**,** and$\psi\in$**B**,U∈**R**， , according to the hypothesis, there must be a triplet <**A**, **R**, **B**> that satisfies the ***Ex*** condition. Similarly, there must be a triplet<**A**, **R**, **B**> that satisfies the ***Uni*** condition.

Delex: Let there is a triplet <**A**, **R**, **B**> generated by <ω, ω′> in $\mathbf{D}_\Phi$. If <U> $\psi\in$**A**, there must be a state ω∈(<U> $\psi)^I$, so that there must be another state ω″ such that <ω, ω″>∈U^I， ω″∉ψ^I. Obviously, <ω, ω″> can produce a triple that satisfies the ***Delex*** condition.

Deluni:Let there is a triplet <**A**, **R**, **B**> generated by <ω, ω′> in $\mathbf{D}_\Phi$. Since [U]$\psi\notin$A, any state ω∉([U]$\psi)^I$, there must be a state ω″ ,such that<ω, ω″>∈U^I， ω″∉ψ^I . Obviously, <ω, ω″> can produce a triple that satisfies the ***Deluni*** condition.

Next, the reconstructed model is symbolized, that is, the reconstructed triplet model $\boldsymbol{D_\Phi}$ is transformed into the corresponding Boolean function, and the SAT decision algorithm is used to determine its satisfiability. Therefore, the

Variable set of the Boolean function ***Var***can be defined as Var $:=[U]\cup CL$ (FLAT ($\boldsymbol{\Phi}$) $\times\{1, 2\}$). The triplets <**A**, **R**, **B**> can be bijective to (**A**$\times$ {1}) ⊔**R**⊔ (**B**$\times$ {2}). Therefore, the concept expression $\varphi$and action U of Iteration-free converse propositional dynamic logic can be expressed by the feature Boolean function, as follows:

$$[\varphi]:=\begin{cases}[<\varphi, 1>] & \text{If } \varphi\in CL(\boldsymbol{\Phi})\\ \neg[\psi] & \text{If } \varphi=\neg\psi\\ [\psi]\wedge[\xi] & \text{If } \varphi=\psi\sqcap\xi\\ [\psi]\vee[\xi] & \text{If } \varphi=\psi\sqcup\xi\end{cases}$$

$[U]:=[V]$ if U=V

[U] represents the Boolean function corresponding to action U. Now, an inference algorithm can be defined based on Boolean functions. The details are as follows:

Algorithm 1 For any iteration - free converse propositional dynamic logic$\boldsymbol{\Phi}$and formula set φ=CL (FLAT($\boldsymbol{\Phi}$)) built on proposition set $\boldsymbol{\Phi_0}$ and assembly $\boldsymbol{\Pi_0}$,Let X be each formula in $\boldsymbol{\Phi}$, then the Boolean function [$\boldsymbol{D_\Phi}$] is constructed based on the following rules:

$$[\boldsymbol{D_\Phi}]_0 := f^{Bc}\wedge f^{Ex}\wedge f^{Uni}$$

$$[\boldsymbol{D_\Phi}]_1 := [\boldsymbol{D_\Phi}]_0\wedge f^{Delex}\wedge f^{Deluni}\wedge f^{Sym}$$

Among:

$$f^{Bc}:=\bigwedge_{\chi\in\Phi}[X]$$

$$f^{Ex}:=\bigwedge_{<U>\varphi\in\varphi}[<\varphi, 2>]\wedge[U]\rightarrow[<<U>\varphi, 1>]$$

$$f^{Uni}:=\bigwedge_{[U]\varphi\in\varphi}[<[U]\varphi, 1>]\wedge[U]\rightarrow[<\varphi, 2>]$$

$$f^{Delex}:=\bigwedge_{<U>\varphi\in\varphi}[<<U>\varphi,1>]\rightarrow[\boldsymbol{D_\Phi}]_0\wedge[U]\wedge[<\varphi, 2>]$$

$$f^{Deluni}:=\bigwedge_{[U]\varphi\in\varphi}[<[U]\varphi,1>]\rightarrow\neg([\boldsymbol{D_\Phi}]_0\wedge[U]\wedge\neg[<\varphi, 2>])$$

$$f^{Sym}(S):=[\boldsymbol{D_\Phi}]_0(\{<\varphi,1>|<\varphi,2>\in V\}\cup\{\text{conv}(R)|\ R\in V\}\cup\{<\varphi,2>|<\varphi,1>\in V\})$$

If [$\boldsymbol{D_\Phi}$](V)=false, for all $V\subseteq$***Var,*** the algorithm returns "unsatisfiable", otherwise it returns "satisfiable".

Because the formula set is limited, the algorithm must be terminated and the determination of the satisfiability of the formula set is correct and effective. Finally, if the Boolean function is known, the satisfiability of the Boolean function can be determined by establishing ordered binary decision diagram(i.e. whether there is a path to 1 from the root node, if so, the Iteration-free converse propositional dynamic logic formula set can be satisfied; Otherwise, the iteration -free converse propositional dynamic logic formula set is not satisfied).

IV. CASE ANYLYSIS

Given an Iteration-free converse propositional dynamic logic formula:

$<\alpha;\beta> (\varphi\wedge\psi)\wedge [\alpha] \varphi$

First, all the formulas are converted into corresponding formulas according to the NCNF paradigm, and FLAT regularization is performed on the formula set. The FLAT (Φ) obtained after the conversion is as follows:

$<\alpha>\xi_1\wedge [\alpha]\varphi\wedge (\neg\xi_1\vee<\beta>\xi_2)\wedge (\neg\xi_2\vee (\varphi\wedge\psi))$

Next, for the FLAT (Φ) obtained in the previous example, apply Algorithm 1 to convert it to a Boolean function, as shown in the following formula:

$f^{Bc}:=<<\alpha>\xi_1,1>\wedge<[\beta]\varphi,1>\wedge(\neg<\xi_1,1>\vee<<\beta>\xi_2,1>)$

$\wedge(\neg<\xi_2,1>\vee(<\varphi,1>\wedge<\psi,1>))$

$f^{Ex}:=(<\xi_1,2>\wedge\alpha\rightarrow<<\alpha>\xi_1,1>)\wedge(<\xi_2,2>\wedge\beta\rightarrow<<\beta>\xi_2,1>)$

$f^{Uni}:=<[\alpha]\varphi,1>\wedge\alpha\rightarrow<\varphi,2>$

$f^{Delex}:=(<<\alpha>\xi_1,1>\rightarrow\alpha\wedge<\xi_1,2>)\wedge(<<\beta>\xi_2,1>\rightarrow\beta\wedge<\xi_2,2>)$

$f^{Deluni}:=<[\alpha]\varphi,1>\rightarrow\neg(\alpha\wedge\neg<\varphi,2>)$

$[\mathbf{D}_\Phi]_0 := f^{Bc}\wedge f^{Ex}\wedge f^{Uni}$

$[\mathbf{D}_\Phi]_1 := [\mathbf{D}_\Phi]_0\wedge f^{Delex}\wedge f^{Deluni}$

Finally, according to the above Boolean expression, the ordered binary decision diagram map corresponding to $[D_\Phi]_1$ is shown in Figure 1, in order to enhance readability. Figure 1 removes all paths to endpoint 0. As can be seen from Figure1, there is an end point of the ordered binary decision diagram that reaches 1, so the corresponding formula set Φ is satisfactory.

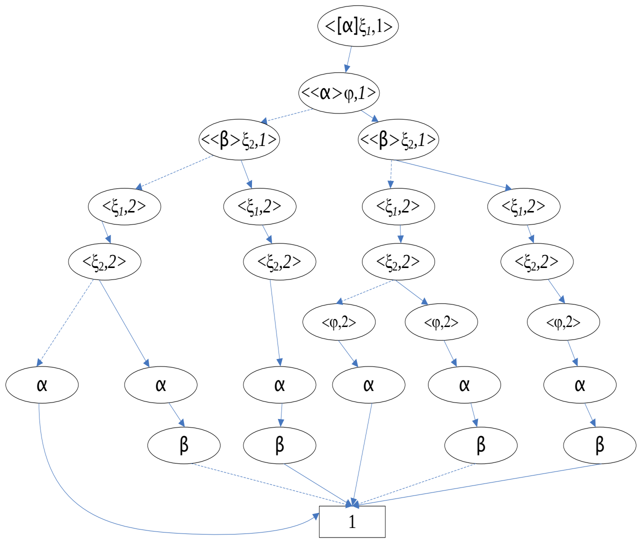

Fig 1OBDD representation of $[\mathbf{D}_{\Phi}]_1$

In order to evaluate the above algorithm, Eclipse was used as the development platform, and the inference engine of converse propositional dynamic logicbased on binary decision graph satisfiability judgment was developed by using javabdd_1.0b2 open source software package. The inference engine can be used to determine the satisfiability of the Iteration-free converse propositional dynamic logic formula set.

V. CONCLUSION

Based on the characteristics of ordered binary decision diagram data structure, this paper gives an algorithm for judging the satisfiability of Iteration-free converse propositional dynamic logic based on ordered binary decision diagram, and proves the termination, reliability and completeness of the algorithm. The algorithm is proved to be correct by a concrete example. The next step is to extend ordered binary decision diagram - based decision algorithm to converse propositional dynamic logic or intersection propositional dynamic logic considering iterative operators and optimize the corresponding algorithm.

REFERENCES

[1] FISCHER M, LADNER R. (1979). Propositional dynamic logic of regular programs. *Computer System Science, 18(2):194-211.*

[2] BAADER F, SATTLER U. (2001). An overview of tableau algorithms for description logic .*Studia Logical,*

3(69):5-40.

[3] BRYANT RE. (1986). Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers, 35(8):677-691.*

[4] PAN GQ, SATTLER U, VARDI MY. (2006). BDD-based decision procedures for the modal logic K.*Journal of Applied Non-Classical Logics, 16(1-2):169-208.*

[5] SCHILD K. (1991).A correspondence theory for terminological logics: *Preliminary report. Proceedings of the 12th on Artificial Intelligence, 466-471.*

[6] RUDOLPH S, KROTZSCH M, HITZLER P. (2008).Description Logic Reasoning with Decision Diagrams-Compiling SHIQ to Disjunctive Datalog.*Proc of the 7th International Semantic Web Conference,435-450.*

[7] PHILIPE BALBIANI, DIMITER VIKARELOV. (2001). Iteration-free PDL with Intersection: a Complete Axiomatization.*Fundamental Informaticae, 4(5):173-194.*

[8] DRECHSLER .R, SIELING.D. Special Section on BDD: Binary Decision Diagrams in Theory and Practice [J]. International Journal on Software Tools for Technology Transfer, 2001, 2(3):112-136.

[9] GU TIANLONG,XUZHOUBO, (2015).Ordered binary decision diagram and application.*Science Press,35-60*

[10] Christian Doczkal, Joachim Bard (2018) Completeness and Decidability of Converse PDL in the Constructive Type Theory of Coq