\def\year{2021}\relax

%File: formatting-instructions-latex-2021.tex

%release 2021.1

\documentclass[letterpaper]{article} % DO NOT CHANGE THIS

\usepackage{aaai21} % DO NOT CHANGE THIS

\usepackage{times} % DO NOT CHANGE THIS

\usepackage{helvet} % DO NOT CHANGE THIS

\usepackage{courier} % DO NOT CHANGE THIS

\usepackage[hyphens]{url} % DO NOT CHANGE THIS

\usepackage{graphicx} % DO NOT CHANGE THIS

\urlstyle{rm} % DO NOT CHANGE THIS

\def\UrlFont{\rm} % DO NOT CHANGE THIS

\usepackage{natbib} % DO NOT CHANGE THIS AND DO NOT ADD ANY OPTIONS TO IT

\usepackage{caption} % DO NOT CHANGE THIS AND DO NOT ADD ANY OPTIONS TO IT

\frenchspacing % DO NOT CHANGE THIS

\setlength{\pdfpagewidth}{8.5in} % DO NOT CHANGE THIS

\setlength{\pdfpageheight}{11in} % DO NOT CHANGE THIS

\usepackage[linesnumbered,boxed]{algorithm2e}

\usepackage{enumerate}

\usepackage{graphicx}

\usepackage{amsmath}

\usepackage{amsthm}

\usepackage{amstext}

\usepackage{amssymb}

%\nocopyright

%PDF Info Is REQUIRED.

% For /Author, add all authors within the parentheses, separated by commas. No accents or commands.

% For /Title, add Title in Mixed Case. No accents or commands. Retain the parentheses.

\pdfinfo{

/Title (AAAI Press Formatting Instructions for Authors Using LaTeX -- A Guide)

/Author (AAAI Press Staff, Pater Patel Schneider, Sunil Issar, J. Scott Penberthy, George Ferguson, Hans Guesgen, Francisco Cruz, Marc Pujol-Gonzalez)

/TemplateVersion (2021.1)

} %Leave this

% /Title ()

% Put your actual complete title (no codes, scripts, shortcuts, or LaTeX commands) within the parentheses in mixed case

% Leave the space between \Title and the beginning parenthesis alone

% /Author ()

% Put your actual complete list of authors (no codes, scripts, shortcuts, or LaTeX commands) within the parentheses in mixed case.

% Each author should be only by a comma. If the name contains accents, remove them. If there are any LaTeX commands,

% remove them.

% DISALLOWED PACKAGES

% \usepackage{authblk} -- This package is specifically forbidden

% \usepackage{balance} -- This package is specifically forbidden

% \usepackage{color (if used in text)

% \usepackage{CJK} -- This package is specifically forbidden

% \usepackage{float} -- This package is specifically forbidden

% \usepackage{flushend} -- This package is specifically forbidden

% \usepackage{fontenc} -- This package is specifically forbidden

% \usepackage{fullpage} -- This package is specifically forbidden

% \usepackage{geometry} -- This package is specifically forbidden

% \usepackage{grffile} -- This package is specifically forbidden

% \usepackage{hyperref} -- This package is specifically forbidden

% \usepackage{navigator} -- This package is specifically forbidden

% (or any other package that embeds links such as navigator or hyperref)

% \indentfirst} -- This package is specifically forbidden

% \layout} -- This package is specifically forbidden

% \multicol} -- This package is specifically forbidden

% \nameref} -- This package is specifically forbidden

% \usepackage{savetrees} -- This package is specifically forbidden

% \usepackage{setspace} -- This package is specifically forbidden

% \usepackage{stfloats} -- This package is specifically forbidden

% \usepackage{tabu} -- This package is specifically forbidden

% \usepackage{titlesec} -- This package is specifically forbidden

% \usepackage{tocbibind} -- This package is specifically forbidden

% \usepackage{ulem} -- This package is specifically forbidden

% \usepackage{wrapfig} -- This package is specifically forbidden

% DISALLOWED COMMANDS

% \nocopyright -- Your paper will not be published if you use this command

% \addtolength -- This command may not be used

% \balance -- This command may not be used

% \baselinestretch -- Your paper will not be published if you use this command

% \clearpage -- No page breaks of any kind may be used for the final version of your paper

% \columnsep -- This command may not be used

% \newpage -- No page breaks of any kind may be used for the final version of your paper

% \pagebreak -- No page breaks of any kind may be used for the final version of your paperr

% \pagestyle -- This command may not be used

% \tiny -- This is not an acceptable font size.

% \vspace{- -- No negative value may be used in proximity of a caption, figure, table, section, subsection, subsubsection, or reference

% \vskip{- -- No negative value may be used to alter spacing above or below a caption, figure, table, section, subsection, subsubsection, or reference

\setcounter{secnumdepth}{0} %May be changed to 1 or 2 if section numbers are desired.

% The file aaai21.sty is the style file for AAAI Press

% proceedings, working notes, and technical reports.

%

% Title

% Your title must be in mixed case, not sentence case.

% That means all verbs (including short verbs like be, is, using,and go),

% nouns, adverbs, adjectives should be capitalized, including both words in hyphenated terms, while

% articles, conjunctions, and prepositions are lower case unless they

% directly follow a colon or long dash

\title{A Resolution Calculus for Forgetting in \CTL}

\author{

%Authors

% All authors must be in the same font size and format.

}

\affiliations{

%Afiliations

% \textsuperscript{\rm 1}Association for the Advancement of Artificial Intelligence\\

%If you have multiple authors and multiple affiliations

% use superscripts in text and roman font to identify them.

%For example,

% Sunil Issar, \textsuperscript{\rm 2}

% J. Scott Penberthy, \textsuperscript{\rm 3}

% George Ferguson,\textsuperscript{\rm 4}

% Hans Guesgen, \textsuperscript{\rm 5}.

% Note that the comma should be placed BEFORE the superscript for optimum readability

% 2275 East Bayshore Road, Suite 160\\

% Palo Alto, California 94303\\

% email address must be in roman text type, not monospace or sans serif

% publications21@aaai.org

% See more examples next

}

\iffalse

%Example, Single Author, ->> remove \iffalse,\fi and place them surrounding AAAI title to use it

\title{My Publication Title --- Single Author}

\author {

% Author

Author Name \\

}

\affiliations{

Affiliation \\

Affiliation Line 2 \\

name@example.com

}

\fi

\iffalse

%Example, Multiple Authors, ->> remove \iffalse,\fi and place them surrounding AAAI title to use it

\title{My Publication Title --- Multiple Authors}

\author {

% Authors

First Author Name,\textsuperscript{\rm 1}

Second Author Name, \textsuperscript{\rm 2}

Third Author Name \textsuperscript{\rm 1} \\

}

\affiliations {

% Affiliations

\textsuperscript{\rm 1} Affiliation 1 \\

\textsuperscript{\rm 2} Affiliation 2 \\

firstAuthor@affiliation1.com, secondAuthor@affilation2.com, thirdAuthor@affiliation1.com

}

\fi

\begin{document}

\newcommand{\tuple}[1]{{\langle{#1}\rangle}}

\newcommand{\Mod}{\textit{Mod}}

\newcommand\ie{{\it i.e. }}

\newcommand\eg{{\it e.g.}}

%\newcommand\st{{\it s.t. }}

\newtheorem{definition}{Definition}

\newtheorem{examp}{Example}

\newenvironment{example}{\begin{examp}\rm}{\end{examp}}

\newtheorem{lemma}{Lemma}

\newtheorem{proposition}{Proposition}

\newtheorem{theorem}{Theorem}

\newtheorem{corollary}[theorem]{Corollary}

%\newenvironment{proof}{{\bf Proof:}}{\hfill\rule{2mm}{2mm}\\ }

\newcommand{\rto}{\rightarrow}

\newcommand{\lto}{\leftarrow}

\newcommand{\lrto}{\leftrightarrow}

\newcommand{\Rto}{\Rightarrow}

\newcommand{\Lto}{\Leftarrow}

\newcommand{\LRto}{\Leftrightarrow}

\newcommand{\Var}{\textit{Var}}

\newcommand{\Forget}{\textit{Forget}}

\newcommand{\KForget}{\textit{KForget}}

\newcommand{\TForget}{\textit{TForget}}

%\newcommand{\forget}{\textit{forget}}

\newcommand{\Fst}{\textit{Fst}}

\newcommand{\dep}{\textit{dep}}

\newcommand{\term}{\textit{term}}

\newcommand{\literal}{\textit{literal}}

\newcommand{\Atom}{\mathcal{A}}

\newcommand{\SFive}{\textbf{S5}}

\newcommand{\MPK}{\textsc{k}}

\newcommand{\MPB}{\textsc{b}}

\newcommand{\MPT}{\textsc{t}}

\newcommand{\MPA}{\forall}

\newcommand{\MPE}{\exists}

\newcommand{\DNF}{\textit{DNF}}

\newcommand{\CNF}{\textit{CNF}}

\newcommand{\degree}{\textit{degree}}

\newcommand{\sunfold}{\textit{sunfold}}

\newcommand{\Pos}{\textit{Pos}}

\newcommand{\Neg}{\textit{Neg}}

\newcommand\wrt{{\it w.r.t.}}

\newcommand{\Hm} {{\cal M}}

\newcommand{\Hw} {{\cal W}}

\newcommand{\Hr} {{\cal R}}

\newcommand{\Hb} {{\cal B}}

\newcommand{\Ha} {{\cal A}}

\newcommand{\Dsj}{\triangledown}

\newcommand{\wnext}{\widetilde{\bigcirc}}

\newcommand{\nex}{\bigcirc}

\newcommand{\ness}{\square}

\newcommand{\qness}{\boxminus}

\newcommand{\wqnext}{\widetilde{\circleddash}}

\newcommand{\qnext}{\circleddash}

\newcommand{\may}{\lozenge}

\newcommand{\qmay}{\blacklozenge}

\newcommand{\unt} {{\cal U}}

\newcommand{\since} {{\cal S}}

\newcommand{\SNF} {\textit{SNF$\_C$}}

\newcommand{\start}{\textbf{start}}

\newcommand{\Elm}{\textit{Elm}}

\newcommand{\simp}{\textbf{simp}}

\newcommand{\nnf}{\textbf{nnf}}

\newcommand{\CTL}{\textrm{CTL}}

\newcommand{\Ind}{\textrm{Ind}}

\newcommand{\Tran}{\textrm{Tran}}

\newcommand{\Sub}{\textrm{Sub}}

\newcommand{\NI}{\textrm{NI}}

\newcommand{\Inst}{\textrm{Inst}}

\newcommand{\Com}{\textrm{Com}}

\newcommand{\Rp}{\textrm{Rp}}

\newcommand{\forget}{{\textsc{f}\_\CTL}}

\newcommand{\ALL}{\textsc{a}}

\newcommand{\EXIST}{\textsc{e}}

\newcommand{\NEXT}{\textsc{x}}

\newcommand{\FUTURE}{\textsc{f}}

\newcommand{\UNTIL}{\textsc{u}}

\newcommand{\GLOBAL}{\textsc{g}}

\newcommand{\UNLESS}{\textsc{w}}

\newcommand{\Def}{\textrm{def}}

\newcommand{\IR}{\textrm{IR}}

\newcommand{\Tr}{\textrm{Tr}}

\newcommand{\dis}{\textrm{dis}}

\def\PP{\ensuremath{\textbf{PP}}}

\def\NgP{\ensuremath{\textbf{NP}}}

\def\W{\ensuremath{\textbf{W}}}

\newcommand{\Pre}{\textrm{Pre}}

\newcommand{\Post}{\textrm{Post}}

\newcommand{\CTLsnf}{{\textsc{SNF}\_{\textsc{ctl}}^g}}

\newcommand{\ResC}{{\textsc{R}\_{\textsc{ctl}}^{\succ, S}}}

\newcommand{\CTLforget}{{\textsc{F}\_{\textsc{ctl}}}}

\newcommand{\Refine}{\textsc{Refine}}

\newcommand{\cf}{\textrm{cf.}}

\newcommand{\NEXP}{\textmd{\rm NEXP}}

\newcommand{\EXP}{\textmd{\rm EXP}}

\newcommand{\coNEXP}{\textmd{\rm co-NEXP}}

\newcommand{\NP}{\textmd{\rm NP}}

\newcommand{\coNP}{\textmd{\rm co-NP}}

\newcommand{\Pol}{\textmd{\rm P}}

\newcommand{\BH}[1]{\textmd{\rm BH}\_{#1}}

\newcommand{\coBH}[1]{\textmd{\rm co-BH}\_{#1}}

\newcommand{\Empty}{\emptyset}%\varnothing}

\newcommand{\NLOG}{\textmd{\rm NLOG}}

\newcommand{\DeltaP}[1]{\Delta\_{#1}^{p}}

\newcommand{\PIP}[1]{\Pi\_{#1}^{p}}

\newcommand{\SigmaP}[1]{\Sigma\_{#1}^{p}}

\newif\ifcolors\colorstrue

% activate next line to remove colors

%\colorsfalse

\ifcolors

\newcommand{\mycomment}[2]{{#1}{{\textcolor{red}{#2}}}}

\else

\newcommand{\mycomment}[2]{#2}

\fi

\maketitle

\begin{abstract}

Computation Tree Logic (\CTL) is a well-known logical formalism in computer science with a wide range of applications; it is used in formal verification in the context of representing and reasoning about high-level system information (or \emph{specification}), but also in other domains e.g., planning. Orthogonal to this, \emph{forgetting} is the field of study which concerns about removing information that is deemed irrelevant or obsolete from a knowledge base in a principled way.

In this paper, we present a resolution-based approach to perform forgetting in \CTL.

%As we show, having such an approach let us perform forgetting while avoiding undesirable complications that arise from handling it on the semantic level.

% More specifically, we develop a calculus which extends earlier work (\CTL\ with \emph{index} i.e., $\CTLsnf$) with additional rules i.e., EF-implication which connects \emph{next\ state} and \emph{future\ state}.

Our technical contribution is manifolded. We provide a bisimulation between \CTL\ formulae and $\CTLsnf$ clauses. We introduce techniques for eliminating undesired atoms. Moreover, we show the termination of our approach and analyse its computational complexity.

\end{abstract}

\section{Introduction}

\noindent Computation Tree Logic (\CTL)~\cite{clarke1981design} is one of the central logical formalisms in computer science with a wide range of applications; it is used mostly in formal verification in the context of representing and reasoning about high-level system information (i.e., represented as specifications), but also in other domains e.g., planning~\cite{giunchiglia1999planning,dal2002planning,akintunde2017planning}.

%In those cases, as the system or planning domain has been updated-some of the elements previously considered are no longer considered-if we have had those specifications, we do not need to reproduce the specifications that different from the original one only in that the latter using lesser atoms. Besides, with the size of the system or planning domain growing, not only the number of available (proposition) increased considerably, but also those specifications are often large in size and becoming more complex. This leads to the specification being difficult to maintain and modify, and costly to reuse for later processing, where only a specific part of an specification is of interest. All of this working directly on the whole of the original specification and building a new sub-specification are inadvisable. Therefore, a strong demand for techniques and automated tools for obtaining the specific sub-specification.

Adopting new goals and properties can cause a system getting more complex in time, and the grow in size can make the verification task prohibitively difficult. Such a disruption can be caused by keeping obsolete --otherwise harmless-- properties, which makes a specification harder to maintain, modify or re-use for later processing.

Therefore, simplifications are required when some of the previously considered components (e.g., atomic variables or subgoals) are no longer required. In such cases, it is vital to modify the specification at hand in such a way that it contains only the relevant vocabulary while it is guaranteed that the role of the remaining sub-components (i.e., sub-specifications) are not disrupted. Hence, techniques and automated tools for safely obtaining a sub-specification (from a specification) based on restricted vocabularies are necessary. This work aims to target such necessity using the notion of \emph{forgetting}~\cite{lin1994forget}; a field of study which concerns about removing high-level information from a knowledge base in a principled way.\footnote{The notion of forgetting has been extended to various logic systems. See~\cite{eiter2019brief} for a recent and comprehensive survey.} In particular, we develop a resolution calculus to compute forgetting in CTL.

%\emph{Forgetting}, a technique for extracting knowledge and ensuring its satisfability, addresses these issues.

%It was first formally defined

%in propostional and first order-logics by Lin and Reiter~\cite{lin1994forget} and extended to other logic systems. %, including the bounded \CTL, i.e. the formulae have a bounded size~\cite{renyansfirstpaper}.

% \emph{Forgetting}, the task of distilling a reduced knowledge base that is relevant to a subset of the signature, addresses these issues.

% As a logical notion, \emph{forgetting} was first formally defined

% in propostional and first order-logics by Lin and Reiter~\cite{lin1994forget} and extended to other logic systems~\cite{eiter2019brief}, including the bounded \CTL, i.e. the formulae have a bounded size~\cite{renyansfirstpaper}.

% Another important property of forgetting is that forgetting some atoms from a given formula do not effect the satisfiability of this formula. This means that if we can eliminate some atoms of a formula in advance, then theoretically speaking, it can speed up the determination of satisfiability to a certain extent. In particular, in the solver \CTL-RP~\footnote{https://sourceforge.net/projects/ctlrp/.}, if we have fewer $\CTLsnf$ clauses, then it will be faster to decide the satisfiability of a formula.

% \emph{Forgetting}, the task of distilling a reduced knowledge base that is relevant to a subset of the signature, addresses these issues.

The forgetting in bounded \CTL (i.e., the formulae with a bounded size), and its theoretical properties have been studied in~\cite{renyansfirstpaper}. Moreover, authors have explored a model-based approach to compute the forgetting under bounded \CTL. It is shown that the result of forgetting a set $V$ of atoms from a \CTL\ $\varphi$ is the disjunction of the characterizing formulas of all the possible model $\Hm$ with $\Hm$ is bisimilar with some model $\Hm'$ of $\varphi$ on $V$ and $\Hm$ is bounded on a size. However, applying this technique to \CTL\ in general is not possible. In this paper, we study the forgetting in \CTL without a restriction. Moreover, as opposed to the model-theoretic approach, we take a syntactic approach i.e., a resolution based approach to compute the forgetting in \CTL.

To come up with such an approach is not a straightforward task, since first of all, the resolution based methods for propositional logic~\cite{lin1994forget,Yisong:2015:arx} and Ackermann-based approach in description logic~\cite{Zhao:2017:IJCAI} require a specific normal form which does not exist in \CTL.

Moreover, while any \CTL\ formula can be transformed into a set of $\CTLsnf$ clauses (which is a variant of \CTL\ ) for which such a normal form does exist, it introduces \emph{indices} and extra atoms which are non-native to standard \CTL. We are addressing both of these challenges.

In particular, we adopt a \emph{Second-Order Quantifier Elimination}~based approach (SOQE)~\cite{gabbay2008second} and extend the resolution calculus in~\cite{zhang2014resolution} to eliminate the elements that need to be forgotten. Moreover, we use a generlised variant of Ackermann's Lemma to

eliminate the atoms introduced in the transformation process.

%by using a \emph{binary bisimulation relation} (one on the set of atoms, one on indices).

%Such a bisimulation relation is an extension of the set-based bisimulation introduced in \cite{renyansfirstpaper} by taking \emph{index} into account.

%The rest of the paper is organised as follows. %Next section reports about the related work.

%In Section 2 we introduce the notation and technical preliminaries.

%In Section 3 we give a more precise definition of the forgetting problem.

%As key contributions, Section 4, introduces the resolution-based approach, as well as the termination and complexity of our algorithm.

%We conclude the paper with some related and future work, as well as a brief discussion. Due to space restrictions and to avoid hindering the flow of content, some of the proves are moved to the supplementary material~\footnote{https://github.com/fengrenyan/Resolution-proof-CTL.git}.

\section{Background} \label{preliminaries}

We start with some technical and notational preliminaries. Throughout this paper we fix a set $\Ha$ of propositional variables (or atoms), and use $V$, $V'$ as subsets of $\Ha$.

% In the following several parts, we will introduce the structure we will use for \CTL, syntax and semantic of \CTL\ and the normal form $\CTLsnf$ (Separated Normal Form with Global Clauses for \CTL) of \CTL~\cite{zhang2009refined}.

\subsection{Kripke structure in \CTL}

In general, a transition system

%\footnote{According to \cite{Baier:PMC:2008},

%a {\em transition system} TS is a tuple $(S, Act,\rto,I, AP, L)$ where

%(1) $S$ is a set of states,

%(2) $\textrm{Act}$ is a set of actions,

%(3) $\rto\subseteq S\times \textrm{Act}\times S$ is a transition relation,

%(4) $I\subseteq S$ is a set of initial states,

%(5) $\textrm{AP}$ is a set of atomic propositions, and

%(6) $L:S\rto 2^{\textrm{AP}}$ is a labeling function.}

can be described by a \emph{Kripke \ structure}\footnote{See~\cite{Baier:PMC:2008} for technical details.} which is a triple $\Hm=(S,R,L)$, where

$S$ is a nonempty set of states, % \footnote{Indeed, every state is identified by a configuration of atoms i.e., which holds in that state.},

$R\subseteq S\times S$, and for each $s\in S$, there

is an $s'\in S$ such that $(s,s')\in R$,

and $L: S\rto 2^{\cal A}$ is a labeling function.

%We call a Kripke structure $\Hm$ on a set $V$ of atoms if $L: S \rto 2^V$, i.e., the labelling function $L$ map every state to $V$ (not the $\Ha$).

Given a Kripke structure $\Hm=(S,R,L)$, a \emph{path} $\pi\_{s\_i}$ starting from $s\_i$ of $\Hm$ is an infinite sequence of states $\pi\_{s\_i}=(s\_i, s\_{i+1} s\_{i+2},\dots)$, where for each $j$ with $0\leq i\leq j$, $(s\_j, s\_{j+1}) \in R$. By $s'\in \pi\_{s\_i}$ we mean that $s'$ is a state occurring in the path $\pi\_{s\_i}$.

A state $s\in S$ is {\em initial} if for any state $s'\in S$, there is a path $\pi\_s$ s.t $s'\in \pi\_s$.

If $s\_0$ is an initial state of $\Hm$, then we denote such a Kripke structure $\Hm$ as $(S,R,L,s\_0)$ and call it an \emph{initial structure}.

% For a given Kripke structure $\Hm=(S,R,L,s\_0)$ and $s\in S$,

% the {\em computation tree}

% $\Tr\_n^{\cal M}(s)$ of $\cal M$ (or simply $\Tr\_n(s)$), that has depth $n \ge 0$ and is rooted at $s$, is recursively defined \cite{DBLP:journals/tcs/BrowneCG88} as follows:

% \begin{itemize}

% \item $\Tr\_0(s)$ consists of a single node $s$ with label $s$.

% \item $\Tr\_{n+1}(s)$ has as its root a node $m$ with label $s$, and

% if $(s,s')\in R$ then the node $m$ has a subtree $\Tr\_n(s')$.

% % \footnote{Though

% % some nodes of the tree may have the same label, they are different nodes in the tree.}.

% \end{itemize}

% %By $s\_n$ we mean a $n$th level node of tree $\Tr\_m(s)$ $(m \geq n)$.

A {\em \MPK-structure} (or {\em \MPK-interpretation}) $\mathcal{K}$ consists of an initial structure

${\cal M}=(S, R, L, s\_0)$ and a state $s\in S$, i.e., $\mathcal{K} = (\mathcal{M}, s)$.

If in addition $s=s\_0$ (i.e., $\mathcal{K} = (\mathcal{M}, s\_0)$), then the \MPK-structure is called an {\em initial} \MPK-structure.

\subsection{Syntax and Semantics of \CTL}

We briefly review the usual basic syntax and semantics

of the \CTL~\cite{DBLP:journals/toplas/ClarkeES86}. For a more elaborate treatment, the reader is invited to standard textbooks such as \cite{Baier:PMC:2008}.

The {\em signature} of the language $\cal L$ of \CTL\ includes:

a finite set $\cal A$ of Boolean variables called {\em atoms}'

constant symbols $\bot$ and $\top$;

the classical connectives $\lor$ and $\neg$;

%\item the propositional constants: $\bot$;

the path quantifiers $\ALL$ and $\EXIST$;

the temporal operators \NEXT, \FUTURE, \GLOBAL\ and \UNTIL, that

means `neXt state', `some Future state', `all future states (Globally)' and `Until', respectively;

and parentheses: "(" and ")".

The priorities for the \CTL\ connectives are the same with that in~\cite{renyansfirstpaper}.

Then the {\em (existential normal form or ENF in short) formulas} of

$\cal L$ are inductively defined via a Backus Naur form:

\begin{equation}\label{def:CTL:formulas}

\phi ::= \bot \mid \top \mid p \mid\neg\phi \mid \phi\lor\phi \mid

\EXIST \NEXT \phi \mid

%\EXIST \FUTURE \phi \mid

\EXIST \GLOBAL \phi \mid

\EXIST (\phi\ \UNTIL\ \phi)%.% \mid

%\ALL \NEXT \phi \mid

% \ALL \FUTURE \phi \mid

% \ALL \GLOBAL \phi \mid

% \ALL [\phi\ \UNTIL\ \phi]

\end{equation}

where $p\in\cal A$. The formulas $\phi\land\psi$ and $\phi\supset \psi$

are defined in the usual way.

Other formulas in $\cal L$ are abbreviated

using the forms in (\ref{def:CTL:formulas}).

For convenience, throughout this article we identify a finite set $\Pi$ of formulas (as the formula $\bigwedge\Pi$) whenever it is clear from the context.

%Notice that, according to the

%above definition for formulas of \CTL,

%each of the \CTL\ {\em temporal connectives} has the form $XY$

%where $X\in \{\ALL,\EXIST\}$ and $Y\in\{\NEXT, \FUTURE, \GLOBAL, \UNTIL\}$.

%The priorities for the \CTL\ connectives are assumed to be (from the highest to the lowest):

%\begin{equation\*}

% \neg, \EXIST\NEXT, \EXIST\FUTURE, \EXIST\GLOBAL, \ALL\NEXT, \ALL\FUTURE, \ALL\GLOBAL

% \prec \land \prec \lor \prec \EXIST\UNTIL, \ALL\UNTIL, \EXIST \UNLESS, \ALL \UNLESS, \rto.

%\end{equation\*}

Next, the usual CTL Semantics is defined as follows.

Let ${\cal M}=(S,R,L,s\_0)$ be a Kripke structure, $s\in S$ and $\phi \in \cal L$.

The {\em satisfiability} relation between $({\cal M},s)$ and $\phi$,

written $({\cal M},s)\models\phi$, is inductively defined on the structure of $\phi$ as follows:

$({\cal M},s)\not\models\bot$ \ and\ $({\cal M},s)\models\top$;

$({\cal M},s)\models p$ iff $p\in L(s)$;

$({\cal M},s)\models \phi\_1\lor\phi\_2$ iff

$({\cal M},s)\models \phi\_1$ or $({\cal M},s)\models \phi\_2$;

$({\cal M},s)\models \neg\phi$ iff $({\cal M},s)\not\models\phi$;

$({\cal M},s)\models \EXIST\NEXT\phi$ iff

$({\cal M},s\_1)\models\phi$ for some $(s,s\_1)\in R$;

$({\cal M},s)\models \EXIST\GLOBAL\phi$ iff

$\cal M$ has a path $(s\_1=s,s\_2,\ldots)$ such that

$({\cal M},s\_i)\models\phi$ for each $i\ge 1$;

$({\cal M},s)\models \EXIST(\phi\_1\UNTIL\phi\_2)$ iff

$\cal M$ has a path $(s\_1=s,s\_2,\ldots)$ such that, for some $i\ge 1$,

$({\cal M},s\_i)\models\phi\_2$ and

$({\cal M},s\_j)\models\phi\_1$ for each $j$ (with $1\leq j<i$)().

Similar to the work in \cite{browne1988characterizing,Bolotov:1999:JETAI},

only initial \MPK-structures are considered to be candidate models

in the following, unless otherwise noted. Formally,

an initial \MPK-structure $\cal K$ is a {\em model} of a formula $\phi$

whenever ${\cal K}\models\phi$.

We denote $\Mod(\phi)$ the set of models of $\phi$.

The formula

$\phi$ is {\em satisfiable}

if $\Mod(\phi)\neq\emptyset$.

Given two formulas $\phi\_1$ and $\phi\_2$, by $\phi\_1\models\phi\_2$ we mean $\Mod(\phi\_1)\subseteq\Mod(\phi\_2)$, by $\phi\_1\equiv\phi\_2$ we mean $\phi\_1\models\phi\_2$ and $\phi\_2\models\phi\_1$.

In this case, $\phi\_1$ is {\em equivalent} to $\phi\_2$.

The set of atoms occurring in $\phi\_1$ is denoted by $\Var(\phi\_1)$.

The formula $\phi\_1$ is {\em irrelevant to} the atoms in a set $V$ (or simply $V$-{\em irrelevant}), written $\IR(\phi\_1,V)$,

if there is a formula $\psi$ with

$\Var(\psi)\cap V=\emptyset$ such that $\phi\_1\equiv\psi$.

% Similar to the work in \cite{DBLP:journals/tcs/BrowneCG88,Bolotov:1999:JETAI},

% only initial \MPK-structures are considered to be candidate models

% in the following, unless otherwise noted. Formally,

% an initial \MPK-structure $\cal K$ is a {\em model} of a formula $\phi$

% whenever ${\cal K}\models\phi$.

% %Let $\Pi$ be a set of formulae, ${\cal K} \models \Pi$ if for each $\phi\in \Pi$ there is $\cal K \models \phi$.

% We denote $\Mod(\phi)$ the set of models of $\phi$.

% The formula $\phi$ is {\em satisfiable}

% iff $\Mod(\phi)\neq\emptyset$.

% Given two formulas $\phi\_1$ and $\phi\_2$, by $\phi\_1\models\phi\_2$, we mean $\Mod(\phi\_1)\subseteq\Mod(\phi\_2)$, and

% by $\phi\_1\equiv\phi\_2$, we mean $\phi\_1\models\phi\_2$ and $\phi\_2\models\phi\_1$.

% In this case, $\phi\_1$ is {\em equivalent} to $\phi\_2$.

% The set of atoms occurring in $\phi\_1$, is denoted by $\Var(\phi\_1)$. We say that

% $\phi\_1$ is $V$-{\em irrelevant}, and write $\IR(\phi\_1,V)$,

% if there is a formula $\psi$ with

% $\Var(\psi)\cap V=\emptyset$ such that $\phi\_1\equiv\psi$.

\subsection{Normal Form in \CTL}

Any \CTL\ formula $\varphi$ can be transformed into a set $T\_\varphi$ of $\CTLsnf$ clauses (Separated Normal Form with Global Clauses for \CTL~\cite{zhang2014resolution}) in polynomial time such that $\varphi$ is satisfiable iff $T\_\varphi$ is satisfiable~\cite{zhang2008first}.

%An important difference between \CTL\ formulae and $\CTLsnf$ is that $\CTLsnf$ is an extension of the syntax of \CTL\ to use indices.

%These indices can be used to preserve a particular path context.

The language of $\CTLsnf$ clauses is defined over an extension of \CTL. That is, the language is based on: (1) the language of CTL; (2) a propositional constant $\start$; (3) a countably infinite index set $\Ind$; and (4) temporal operators: $\EXIST\_{\tuple{ind}} \NEXT$, $\EXIST\_{\tuple{ind}} \FUTURE$, $\EXIST\_{\tuple{ind}} \GLOBAL$, $\EXIST\_{\tuple{ind}} \UNTIL$, and $\EXIST\_{\tuple{ind}} \UNLESS$ with $ind \in \Ind$.

%The priorities for the $\CTLsnf$\ connectives are assumed to be (from the highest to the lowest):

%\begin{align\*}

% &\neg, (\EXIST\NEXT,\EXIST\_{\tuple{ind}}\NEXT), (\EXIST\FUTURE ,\EXIST\_{\tuple{ind}}\FUTURE), (\EXIST\GLOBAL,\EXIST\_{\tuple{ind}} \GLOBAL), \ALL\NEXT, \ALL\FUTURE, \ALL\GLOBAL \\

% &\prec \land \prec \lor \prec (\EXIST\UNTIL,\EXIST\_{\tuple{ind}} \UNTIL), \ALL\UNTIL, (\EXIST \UNLESS, ,\EXIST\_{\tuple{ind}}\UNLESS), \ALL \UNLESS, \rto.

%\end{align\*}

%Where the operators in the same brackets have the same priority.

%The $\CTLsnf$ clauses consists of formulae of the following forms: $\ALL \GLOBAL(\start \supset \bigvee\_{j=1}^k m\_j)$ (initial clause), $\ALL \GLOBAL(true \supset \bigvee\_{j=1}^k m\_j)$ (global clause), $\ALL \GLOBAL(\bigwedge\_{i=1}^n l\_i \supset \ALL \NEXT \bigvee\_{j=1}^k m\_j)$ (\ALL-step clause), $\ALL \GLOBAL(\bigwedge\_{i=1}^n l\_i \supset \EXIST\_\tuple{ind} \NEXT \bigvee\_{j=1}^k m\_j)$ (\EXIST-step clause), $\ALL \GLOBAL(\bigwedge\_{i=1}^n l\_i \supset \ALL \FUTURE l)$ (\ALL-sometime clause) and $\ALL \GLOBAL(\bigwedge\_{i=1}^n l\_i \supset \EXIST\_{\tuple{ind}} \FUTURE l)$ (\EXIST-sometime clause),

%We introduce the $\CTLsnf$ clauses at first, and then we will talk about its semantics.

More specifically, a $\CTLsnf$ clause is a formula in one of the following forms.

\begin{align\*}

& \ALL \GLOBAL(\start \supset \bigvee\_{j=1}^k m\_j) && (initial\ clause) \\

& \ALL \GLOBAL(true \supset \bigvee\_{j=1}^k m\_j) && (global\ clause) \\

& \ALL \GLOBAL(\bigwedge\_{i=1}^n l\_i \supset \ALL \NEXT \bigvee\_{j=1}^k m\_j) && (\ALL-step\ clause)\\

& \ALL \GLOBAL(\bigwedge\_{i=1}^n l\_i \supset \EXIST\_\tuple{ind} \NEXT \bigvee\_{j=1}^k m\_j) && (\EXIST-step\ clause)\\

& \ALL \GLOBAL(\bigwedge\_{i=1}^n l\_i \supset \ALL \FUTURE l) && (\ALL-sometime\ clause)

\end{align\*}

\begin{align\*}

& \ALL \GLOBAL(\bigwedge\_{i=1}^n l\_i \supset \EXIST\_{\tuple{ind}} \FUTURE l) && (\EXIST-sometime\ clause).

\end{align\*}

where $k \ge 0$, $n > 0$, $\start$ is a propositional constant, $l\_i$ (with $1 \le i \le n$), $m\_j$ (with $1 \le j \le k$) and $l$ are literals, that is, atomic propositions or their negation, and $ind\in \Ind$. %(Ind is a countably infinite index set).

By a clause, we mean the classical clause or the $\CTLsnf$ clause unless explicitly stated.

As all clauses are of the form $\ALL \GLOBAL(P \supset D)$ , we often simply write $P \supset D$ instead.

%A set $T$ of $\CTLsnf$ clauses is satisfiable if there is a model $\Hm=(S, R, L, [\\_], s\_0)$ \st\ for all clause $C\in T$, $(\Hm, s\_0) \models C$.

Formulae of $\CTLsnf$ over $\Ha$ are interpreted in \Ind-Kripke structure $\Hm=(S,R,L, [\\_], s\_0)$, where $S$, $R$, $L$ and $s\_0$ is the same as in aforementioned initial structure and $[\\_]: \Ind \rto 2^{(S\times S)}$ maps every index $ind \in \Ind$ to a successor function $[ind]$ such that for every $s\in S$ there exists exactly a state $s'\in S$ such that $(s,s')\in [ind] \cap R$.

%In this paper we do not need a strict tree Kripke structure as in~\cite{zhang2009refined}, that is we do not those restrictions on $s\_0$ due to that only for simplifying the proof but do not impact the satisfiability of a formula~\cite{zhang2009refined}.

A path $\pi\_{s\_i}^{\tuple{ind}}$ is an infinite sequence of states $s\_i, s\_{i+1}, s\_{i+2},\dots$ such that for every $j\geq i$, $(s\_j, s\_{j+1})\in [ind]$.

%https://www.overleaf.com/project/5f3bcc3285156d0001b683f9

%Let $\pi$ be a path in \Ind-Kripke structure $\Hm$, by $s\in \pi$ we mean that $s$ is a state in the path $\pi$.

Similarly, an {\em \Ind-structure} (or {\em \Ind-interpretation}) is an \Ind-Kripke structure

${\cal M}=(S, R, L, [\\_], s\_0)$ associated

with a state $s\in S$, which we simplify in short as $({\cal M},s)$ in the following.

In case that $s$ is an initial state of $\cal M$, the \Ind-structure is {\em initial}.

%The semantics of $\CTLsnf$ is an extension of the semantics of \CTL\ defined in Section 2.2 except using the \Ind-Kripke structure $\Hm=(S,R,L,[\\_],s\_0)$ replace Kripke structure, $({\cal M},s\_i) \models \start$ iff $s\_i=s\_0$ and for all $\EXIST\_{\tuple{ind}} \Gamma$ are explained in the path $\pi\_{s\_i}^{\tuple{ind}}$, where $\Gamma\in \{\NEXT, \GLOBAL, \UNTIL,\UNLESS\}$.

The semantics of $\CTLsnf$ is then

defined as an extension of the \CTL\ semantics.

% Let $\varphi$ and $\psi$ be two $\CTLsnf$ formulae, $\Hm=(S,R,L,[\\_],s\_0)$ be an \Ind-Kripke structure and $s\_i \in S$. The relation ``$\models$" between $\CTLsnf$ formulae and $(\Hm,s\_i)$ is defined recursively as follows:

Let ${\cal M}=(S,R,L,[\\_],s\_0)$ be an \Ind-Kripke structure, $s\in S$ and $\psi$ a $\CTLsnf$ formulae.

The {\em satisfiability} relation between $({\cal M},s)$ and $\psi$,

written as $({\cal M},s)\models\psi$, is inductively defined on the structure of $\psi$ as follows:

\begin{itemize}

\item $({\cal M},s) \models \start$ iff $s=s\_0$;

\item $({\cal M},s)\models \EXIST\_{\tuple{ind}} \NEXT \psi$ iff for the path $\pi\_{s}^{\tuple{ind}}$, $(\Hm, s')\models \psi$ with $(s, s') \in [ind]$;

\item $({\cal M},s)\models \EXIST\_{\tuple{ind}}\GLOBAL\psi$ iff

for every $s'$ occurring in $\pi\_{s}^{\tuple{ind}}$,

$(\Hm,s') \models \psi$;

\item $({\cal M},s)\models \EXIST\_{\tuple{ind}}[\psi\_1\UNTIL\psi\_2]$ iff

there exists $s'$ occurring in $ \pi\_{s}^{\tuple{ind}}$ such that $(\Hm,s') \models \psi\_2$ and for every $s\_k \in \pi\_{s}^{\tuple{ind}}$, if $i\leq k < j$, then $(\Hm,s\_k) \models \psi\_1$;

\item $(\Hm,s) \models \EXIST\_{\tuple{ind}} \FUTURE \psi$ iff $(\Hm,s) \models \EXIST\_{\tuple{ind}}[\top \UNTIL\psi]$.

\item $({\cal M},s)\models \EXIST\_{\tuple{ind}}[\varphi\UNLESS\psi]$ iff $(\Hm,s) \models \EXIST\_{\tuple{ind}}\GLOBAL \varphi$ or $({\cal M},s)\models \EXIST\_{\tuple{ind}}[\varphi\UNTIL\psi]$.

\end{itemize}

The semantics of the remaining operators are analogous to that of \CTL\ given previously.

%but in the extended \Ind-Kripke structure ${\cal M}=(S, R, L, [\\_],s\_0)$.

A $\CTLsnf$ formula $\varphi$ is satisfiable, iff for some \Ind-Kripke structure $\Hm=(S,R,L,[\\_],s\_0)$, $(\Hm,s\_0)\models \varphi$, and unsatisfiable otherwise. Furthermore, if $(\Hm,s\_0)\models \varphi$ then $(\Hm,s\_0)$ is called an \Ind-model of $\varphi$, and we say that $(\Hm,s\_0)$ satisfies $\varphi$.

By $T \wedge \varphi$, we mean $\bigwedge\_{\psi\in T} \psi \wedge \varphi$, where $T$ is a finite set of formulae.

Other terminologies are similar to those in \CTL\ subsection.

\subsection{Semantic Forgetting in \CTL}\label{problem}

% In this section we will introduce forgetting based on the notion of $V$-bisimulation.

% Intuitively, bisimulation relates states that mutually mimic all individual transitions~\cite{Baier:PMC:2008}. While the $V$-bisimulation relates states that mutually mimic all individual transitions on $\Ha - V$, this is close with forgetting some set $V$ of atoms.

The concept of semantic forgetting in bounded \CTL\ is formally introduced in a recent work~\cite{renyansfirstpaper}. In this part, we shall borrow few notions from that work and also following it, we will give a formal definition of the problem of forgetting in \CTL.

The notion of \empj{bisimulation} captures the idea that two structures are behaviourally the same.

In forgetting, one needs to generalize this idea under some set of $V$; a bisimulation w.r.t. different sets of atomic variables explicitly under a single setting \cite{Yan:AIJ:2009}. One such notion, called $V$-\emph{bisimulation}, between two $\MPK$-structures is introduced in \cite{renyansfirstpaper}. For our purpose, we shall borrow this notion and alter it to $V$-bisimulation between two $\Ind$-structures, which has exactly the same intuition.

First, we start with some useful notation:

$\Hm'=(S',R',L',s\_0')$, $\Hm\_i=(S\_i, R\_i,L\_i, s\_0^i)$ (or

$\Hm'=(S',R',L', [\\_],s\_0')$, $\Hm\_i=(S\_i, R\_i,L\_i, [\\_], s\_0^i)$) and ${\cal K}\_i=(\Hm\_i, s\_i)$ with $s\_i \in S\_i$

where $i \in \mathbb{N}$.

Following \cite{renyansfirstpaper}, we introduce the following construction; intuitively, $V$-bisimulation up to a certain degree (of depth) $n \in \mathbb{N}$ (denoted by $\Hb\_n$).

Let ${\cal K}\_i=({\cal M}\_i,s\_i)$ with $i\in\{1,2\}$. Then, $({\cal K}\_1,{\cal K}\_2)\in\Hb\_0$ if $L\_1(s\_1)- V=L\_2(s\_2)- V$; for $n\ge 0$, $({\cal K}\_1,{\cal K}\_2)\in\Hb\_{n+1}$ if: (1) $({\cal K}\_1,{\cal K}\_2)\in\Hb\_0$, and (2) for every $(s\_i,s\_i')\in R\_i$, there is a $(s\_{(i \mod 2)+1}, s\_{(i \mod 2)+1}')\in R\_{(i \mod 2)+1})$

such that $({\cal K}\_1',{\cal K}\_2')\in \Hb\_n$ where ${\cal K}\_i'=({\cal M}\_i,s\_i')$ with $i\in\{1,2\}$.

% \begin{itemize}

% \item $({\cal K}\_1,{\cal K}\_2)\in\Hb\_0$ if $L\_1(s\_1)- V=L\_2(s\_2)- V$; % and ${\cal K}'=(\tuple{S', R',L'},s')$;

% \item for $n\ge 0$, $({\cal K}\_1,{\cal K}\_2)\in\Hb\_{n+1}$ if:

% \begin{itemize}

% \item $({\cal K}\_1,{\cal K}\_2)\in\Hb\_0$,

% \item for every $(s\_i,s\_i')$ occurring in $R\_i$, there is a $(s\_{(i \mod 2)+1}, s\_{(i \mod 2)+1}')\in R\_{(i \mod 2)+1})$

% such that $({\cal K}\_1',{\cal K}\_2')\in \Hb\_n$,

% % \item for every $(s\_2,s\_2')\in R\_2$, there is a $(s\_1,s\_1')\in R\_1$

% % such that $({\cal K}\_1',{\cal K}\_2')\in \Hb\_n$,

% \end{itemize}

% where ${\cal K}\_i'=({\cal M}\_i,s\_i')$ with $i\in\{1,2\}$.

% \end{itemize}

%Now, we define the notion of $V$-bisimulation between \MPK-structures (\Ind-structures):

\begin{definition}[$V$-bisimulation \cite{renyansfirstpaper}] %~\cite{renyansfirstpaper}

\label{def:V-bisimulation}

Let $V\subseteq\cal A$. Two \MPK-structures (or \Ind-structures) ${\cal K}\_1$ and ${\cal K}\_2$ are $V$-{\em bisimilar}, denoted ${\cal K}\_1 \lrto\_V {\cal K}\_2$,

if and only if $({\cal K}\_1,{\cal K}\_2)\in {\Hb\_i}\mbox{ for all }i\ge 0.$ Moreover, two paths $\pi\_i=(s\_{i,1},s\_{i,2},\ldots)$ of $\Hm\_i$ with $i\in \{1,2\}$

are $V$-{\em bisimilar} if

$ {\cal K}\_{1,j} \lrto\_V {\cal K}\_{2,j}\mbox { for every $j\ge 1$ }$

where ${\cal K}\_{i,j}=(\Hm\_i,s\_{i,j})$.

\end{definition}

Notice that we have made a very small addition to the original definition: namely, defining it between \Ind-structures. Such addition is straightforward.

%It is apparent that $\lrto\_V$ is a binary relation.

In the sequel, we abbreviate ${\cal K}\_1 \lrto\_V {\cal K}\_2$

by $s\_1 \lrto\_V s\_2 $

whenever the underlying initial structures (\Ind-Kripke structures) of states $s\_1$ and $s\_2$ are clear from the context.

% In order to define our problem, \ie forgetting in \CTL, we give the definition of $V$-bisimulation at first. %(read ?? for more detials).

% \begin{definition}\label{def:Vbi}

% Let $V\subseteq\cal A$

% %${\cal M}\_i=(S\_i,R\_i,L\_i,s\_0^i)~(i=1,2)$ be Kripke structures

% and ${\cal K}\_i=({\cal M}\_i,s\_i)~(i=1,2)$ be \MPK-structures (Ind-structures).

% Then $({\cal K}\_1,{\cal K}\_2)\in\cal B$ if and only if

% \begin{enumerate}[(i)]

% \item $L\_1(s\_1)- V = L\_2(s\_2)-V$,

% \item for every $(s\_1,s\_1')\in R\_1$, there is $(s\_2,s\_2')\in R\_2$

% such that $({\cal K}\_1',{\cal K}\_2')\in \Hb$, and

% \item for every $(s\_2,s\_2')\in R\_2$, there is $(s\_1,s\_1')\in R\_1$

% % such that $({\cal K}\_1',{\cal K}\_2')\in \Hb$,

% \end{enumerate}

% where ${\cal K}\_i'=({\cal M}\_i,s\_i')$ with $i\in\{1,2\}$.

% \end{definition}

% In the sequel, we abbreviate ${\cal K}\_1 \lrto\_V {\cal K}\_2$

% by $s\_1 \lrto\_V s\_2 $

% whenever the underlying Kripke structures of states $s\_1$ and $s\_2$ are clear from the context.

% \begin{lemma}\label{lem:equive}~\cite{renyansfirstpaper}

% The relation $\lrto\_V$ is an equivalence relation.

% \end{lemma}

%Besides, we have the following properties:

% It's shown in Proposition 1 of~\cite{renyansfirstpaper} that if a \MPK-structure (or an Ind-structure) is $V\_1$ and $V\_2$-bisimilar with the other two \MPK-structures (or Ind-structures) respectively, then those two \MPK-structures (or Ind-structures) are $V\_1 \cup V\_2$-bisimilar. This is important for forgetting since this laid the foundation of resolving atoms in $V$ one by one in the resolution process later.

% Moreover, the $V\_1$-bisimulation between two \MPK-structures (Ind-structures) implies that these two \MPK-structure (Ind-structures) are $V\_2$-bisimilar for each $V\_2$ with $V\_1 \subseteq V\_2 \subseteq \Ha$.

% %Formally,

% \begin{proposition}\label{div}~\cite{renyansfirstpaper}

% Let $i\in \{1,2\}$, $V\_1,V\_2\subseteq\cal A$

% %$s\_i'$s be two states and

% % $\pi\_i'$s be two pathes,

% and ${\cal K}\_i=({\cal M}\_i,s\_i)~(i=1,2,3)$ be \MPK-structures (Ind-structures)

% such that

% ${\cal K}\_1\lrto\_{V\_1}{\cal K}\_2$ and ${\cal K}\_2\lrto\_{V\_2}{\cal K}\_3$.

% Then:

% \begin{enumerate}[(i)]

% % \item $s\_1'\lrto\_{V\_i}s\_2'~(i=1,2)$ implies $s\_1'\lrto\_{V\_1\cup V\_2}s\_2'$;

% % \item $\pi\_1'\lrto\_{V\_i}\pi\_2'~(i=1,2)$ implies $\pi\_1'\lrto\_{V\_1\cup V\_2}\pi\_2'$;

% % \item for each path $\pi\_{s\_1}$ of $\Hm\_1$ there is a path $\pi\_{s\_2}$ of $\Hm\_2$ such that $\pi\_{s\_1} \lrto\_{V\_1} \pi\_{s\_2}$, and vice versa;

% \item ${\cal K}\_1\lrto\_{V\_1\cup V\_2}{\cal K}\_3$;

% \item If $V\_1 \subseteq V\_2$ then ${\cal K}\_1 \lrto\_{V\_2} {\cal K}\_2$.

% \end{enumerate}

% \end{proposition}

% Intuitively, if two \MPK-structures are $V$-bisimilar, then they satisfy the same formula $\varphi$ that does not contain any atoms in $V$, \ie $\IR(\varphi, V)$.

% \begin{theorem}\label{thm:V-bisimulation:EQ}~\cite{renyansfirstpaper}

% Let $V\subseteq\cal A$, ${\cal K}\_i~(i=1,2)$ be two \MPK-structures such that

% ${\cal K}\_1\lrto\_V{\cal K}\_2$ and $\phi$ a formula with $\IR(\phi,V)$. Then

% ${\cal K}\_1\models\phi$ if and only if ${\cal K}\_2\models\phi$.

% \end{theorem}

% \begin{proof}(sketch) See~\cite{renyansfirstpaper}.

% This can be proved by induction on the structures of $\phi$. % and supposing $\Var(\phi) \cap V = \Empty$ due to $\IR(\phi,V)$.

% For instance, let $\phi = \psi\_1 \vee \psi\_2$, the induction hypothesis is ${\cal K}\_1 \models \psi\_i$ iff ${\cal K}\_2 \models \psi\_i$ with $i\in \{1,2\}$. Then we can see that ${\cal K}\_1 \models \phi$ iff ${\cal K}\_1 \models \psi\_1$ or ${\cal K}\_1 \models \psi\_2$ iff ${\cal K}\_2 \models \psi\_1$ or ${\cal K}\_2 \models \psi\_2$ by induction hypothesis.

% %Other cases can be proved similarly.

% \end{proof}

%Now we give the formal definition of forgetting in \CTL\ from the semantic forgetting point view.

Using $V$-bisimulation, the notion of forgetting in \CTL\ is defined as follows~\cite{renyansfirstpaper}.

% This means that the result of forgetting the atoms in the set $V$ of atoms from \CTL\ formula $\varphi$ is a formula which shares the same models as $\varphi$ and models that are $V$-bisimilar to one of the models of $\varphi$.

\begin{definition}[Forgetting \cite{renyansfirstpaper}]\label{def:V:forgetting}

Let $V\subseteq\cal A$ and $\phi$ a \CTL\ formula.

A \CTL\ formula $\psi$ with $\Var(\psi)\cap V=\emptyset$

is a {\em result of forgetting $V$ from} $\phi$ (denoted as $\CTLforget(\phi,V)$), if

\begin{equation\*}

\Mod(\psi)=\{{\cal K}\mbox{ is initial}\mid \exists {\cal K}'\in\Mod(\phi)\ \&\ {\cal K}'\lrto\_V{\cal K}\},

\end{equation\*}

where $\cal K$ and ${\cal K}'$ are $\MPK$-structures.

\end{definition}

Note that if both $\psi$ and $\psi'$ are results of forgetting $V$ from $\phi$ then

$\Mod(\psi)=\Mod(\psi')$, \ie, $\psi$ and $\psi'$ have the same models. In other words, the forgetting result is unique (up to equivalence).

\section{The Calculus}

Resolution (calculus) is a well-known rule of inference that has been widely studied and applied in automated theorem proving in propositional logic and first-order logic~\cite{leitsch2012resolution}. In the context of \CTL\ , it has been used to decide the satisfiability of a formula \cite{bolotov2000clausal,zhang2014resolution}.

In this section, we shall extend and modify the framework which is used for $\CTLsnf$ in \cite{zhang2014resolution} to compute forgetting in \CTL\ which is not a trivial task and indeed the contribution of this article.

The first key challenge we have to confront is \emph{how to bridge the gap between} \CTL\ \emph{and $\CTLsnf$?} (This is, in particular, necessary since there are indices for existential quantifiers in $\CTLsnf$ e.g., see Table~1 and Table~2).

In order to overcome this problem and compute the forgetting in \CTL\ eventually, %which uses indices for existential quantifiers,

%Similar with the $V$-bisimulation between \MPK-structures,

we need to introduce an auxiliary model-theoretic notion that is $\tuple{V,I}$-bisimulation between \Ind-structures as follows:

\begin{definition}[Binary bisimulation relation] \label{def:VInd:bisimulation}

%\textbf{($\tuple{V,I}$-bisimulation)}

Let $\Hm\_i=(S\_i, R\_i, L\_i, [\\_]\_i, s\_0^i)$ with $i\in \{1, 2\}$ be two \Ind-structures, $V$ be a set of atoms and $I \subseteq Ind$. The $\tuple{V,I}$-bisimulation $\beta\_{\tuple{V,I}}$ between initial \Ind-structures is a set that satisfy $((\Hm\_1, s\_0^1), (\Hm\_2, s\_0^2)) \in \beta\_{\tuple{V,I}}$ if and only if $(\Hm\_1, s\_0^1) \lrto\_V (\Hm\_2, s\_0^2)$ and $\forall j \in (\Ind-I)$ there is: for each $(s, s\_1)\in [j]\_i$ there exists $(s',s\_1')\in [j]\_{(i\mod 2)+1}$ such that $s\lrto\_V s'$ and $s\_1 \lrto\_V s\_1'$.

% \begin{enumerate}[ ]

% \item for each $(s, s\_1)\in [j]\_i$ there exists $(s',s\_1')\in [j]\_{(i\mod 2)+1}$ such that $s\lrto\_V s'$ and $s\_1 \lrto\_V s\_1'$.

% % \item for each $(s', s\_1')\in [j]\_2$ there exists $(s,s\_1)\in [j]\_1$ such that $s\lrto\_V s'$ and $s\_1 \lrto\_V s\_1'$.

% \end{enumerate}

%$\forall j \notin I$ there is $[j]\_1 = [j]\_2$.

\end{definition}

We call this notion as \emph{binary bisimulation relation}, also denoted as $\lrto\_{\tuple{V,I}}$. The intuition is similar to the notion of $V$-bisimulation except that $\tuple{V,I}$-bisimulation takes index into account.

Clearly, $\lrto\_{\tuple{V,I}}$ will degenerate into $\lrto\_V$ when the considered formula is in \CTL\ since we do not need to consider the index anymore. That is, for any two $\Hm\_i=(S\_i, R\_i, L\_i, [\\_]\_i, s\_0^i)$ with $i\in \{1, 2\}$, if $(\Hm\_1, s\_0^1) \lrto\_{\tuple{V,I}} (\Hm\_2, s\_0^2)$, then we have $(\Hm\_1', s\_0^1) \lrto\_V (\Hm\_2', s\_0^2)$ under \CTL where $\Hm\_i'=(S\_i, R\_i, L\_i, s\_0^i)$ with $i\in \{1, 2\}$. Inspired by Proposition 1 in~\cite{renyansfirstpaper}, we have the following result in which indices are taken into account.

%This new type of bisimulation will later be used to show the \emph{equivalence} between a \CTL\ formula and a $\CTLsnf$ formula.

%Besides, it is not difficult to prove $\tuple{V,I}$-bisimulation possess those properties (talked-above) possessed by $V$-bisimulation.

% It's shown in Proposition 1 of~\cite{renyansfirstpaper} that if a \MPK-structure (or an Ind-structure) is $V\_1$ and $V\_2$-bisimilar with the other two \MPK-structures (or Ind-structures) respectively, then those two \MPK-structures (or Ind-structures) are $V\_1 \cup V\_2$-bisimilar. This is important for forgetting since this laid the foundation of resolving atoms in $V$ one by one in the resolution process later.

% Moreover, the $V\_1$-bisimulation between two \MPK-structures (Ind-structures) implies that these two \MPK-structure (Ind-structures) are $V\_2$-bisimilar for each $V\_2$ with $V\_1 \subseteq V\_2 \subseteq \Ha$.

% %Formally,

\begin{proposition}\label{pro:VI:div}

Let $i\in \{1,2\}$, $V\_1,V\_2\subseteq\cal A$, $I\_1, I\_2 \subseteq \Ind$

and ${\cal K}\_i=({\cal M}\_i,s\_0^i)~(i=1,2,3)$ be initial Ind-structures

such that

${\cal K}\_1\lrto\_{\tuple{V\_1, I\_1}}{\cal K}\_2$ and ${\cal K}\_2\lrto\_{\tuple{V\_2,I\_2}}{\cal K}\_3$.

Then:

\begin{enumerate}[(i)]

% \item $s\_1'\lrto\_{V\_i}s\_2'~(i=1,2)$ implies $s\_1'\lrto\_{V\_1\cup V\_2}s\_2'$;

% \item $\pi\_1'\lrto\_{V\_i}\pi\_2'~(i=1,2)$ implies $\pi\_1'\lrto\_{V\_1\cup V\_2}\pi\_2'$;

% \item for each path $\pi\_{s\_1}$ of $\Hm\_1$ there is a path $\pi\_{s\_2}$ of $\Hm\_2$ such that $\pi\_{s\_1} \lrto\_{V\_1} \pi\_{s\_2}$, and vice versa;

\item ${\cal K}\_1\lrto\_{\tuple{V\_1\cup V\_2, I\_1 \cup I\_2}}{\cal K}\_3$;

\item If $V\_1 \subseteq V\_2$ and $I\_1 \subseteq I\_2$ then ${\cal K}\_1 \lrto\_{\tuple{V\_2, I\_2}} {\cal K}\_2$.

\end{enumerate}

\end{proposition}

% \begin{proof}

% %This can be proved similarly with Proposition~\ref{div}.

% (i) By Proposition~\ref{div} we have ${\cal K}\_1\lrto\_{V\_1\cup V\_2}{\cal K}\_3$. For (i) of Definition~\ref{def:VInd:bisimulation} we can prove it as follows:

% for all $(s,s\_1) \in [j]\_1$ there is a $(s', s\_1') \in [j]\_2$ such that $s\lrto\_{V\_1} s'$ and $s\_1 \lrto\_{V\_1} s\_1'$ and there is a $(s'', s\_1'') \in [j]\_3$ such that $s'\lrto\_{V\_2} s''$ and $s\_1' \lrto\_{V\_2} s\_1''$, then we have for all $(s,s\_1) \in [j]\_1$ there is a $(s'', s\_1'') \in [j]\_3$ such that $s \lrto\_{V\_1\cup V\_2} s''$ and $s\_1 \lrto\_{V\_1\cup V\_2} s\_1''$. The (ii) of Definition~\ref{def:VInd:bisimulation} can be proved similarly.

% (ii) This can be proved from (ii) of Proposition~\ref{div}.

% \end{proof}

%Apparently, this Proposition contain the same meaning with Proposition~\ref{div} except the index have been take into consider in this result.

We are now ready to introduce our resolution calculus for forgetting. We shall work with the transformation rules Trans(1) to Trans(8), Trans(10) and Trans(11) in Table~\ref{tab:trans} and resolution rules (SRES1), \dots, (SRES8), (RW1), (RW2), (ERES1), (ERES2) in Table~\ref{tab:res}, which are from~\cite{zhang2014resolution}. Another key challenge one has to confront is the \emph{elimination of the irrelevant atoms (from both those we want to forget and those that introduced by these rules)} which has no straightforward solution.

To provide with a solution for this problem, we shall eventually introduce a new \emph{elimination} operator.

\begin{table\*}[]

\centering

\footnotesize

\begin{tabular}{l l}

\hline

$\textbf{Trans(1)}\ q \supset \EXIST T \varphi \rto q\supset \EXIST\_{\tuple{ind}} T \varphi$ & \textbf{Trans(2)}\ $q \supset \EXIST (\varphi\_1 \UNTIL \varphi\_2) \rto q\supset \EXIST\_{\tuple{ind}} (\varphi\_1 \UNTIL \varphi\_2)$ \\

\textbf{Trans(3)}\ $$ q\supset \varphi\_1 \wedge \varphi\_2 \rto \left\{

\begin{aligned}

& q\supset \varphi\_1 \\

& q\supset \varphi\_2

\end{aligned}

\right.

$$ &

\textbf{Trans(4)}

$$ q\supset \varphi\_1 \vee \varphi\_2 \rto \left\{

\begin{aligned}

& q\supset \varphi\_1 \vee p, && \hbox{if $\varphi\_2$ is not a disjunct}\\

& p\supset \varphi\_2

\end{aligned}

\right.

$$ \\

\textbf{Trans(5)}

$$ \left\{

\begin{aligned}

& q\supset D \rto \top \supset \neg q \vee D\\

& q\supset \perp\rto \top \supset \neg q \\

& q \supset \top \rto \{\}

\end{aligned}

\right.

$$ & \textbf{Trans(6)}

$$ q\supset Q\NEXT \varphi \rto \left\{

\begin{aligned}

& q\supset Q\NEXT p, && \hbox{if $\varphi$ is not a disjunct}\\

& p\supset \varphi

\end{aligned}

\right.

$$\\

\textbf{Trans(7)}

$$ q\supset Q\FUTURE \varphi\rto \left\{

\begin{aligned}

& q\supset Q\FUTURE p, && \hbox{if $\varphi$ is not}\\

& \qquad \hbox{a literal}\\

& p\supset \varphi

\end{aligned}

\right.

$$& \textbf{Trans(8)} $$ q\supset Q(\varphi\_1 \UNTIL \varphi\_2) \rto \left\{

\begin{aligned}

& q\supset Q(\varphi\_1 \UNTIL p), && \hbox{if $\varphi\_2$ is not}\\

& \qquad \hbox{a literal} \\

& p\supset \varphi\_2,

\end{aligned}

\right.$$ \\

\textbf{Trans(10)} $$ q\supset Q\GLOBAL \varphi \rto \left\{

\begin{aligned}

& q \supset p\\

& p\supset \varphi\\

& p\supset Q\NEXT p

\end{aligned}

\right.

$$ & \textbf{Trans(11)} $$ q\supset Q(\varphi \UNTIL l) \rto \left\{

\begin{aligned}

& q \supset l\vee p\\

& p\supset \varphi\\

& p\supset Q\NEXT(l\vee p)\\

& q\supset Q \FUTURE l

\end{aligned}

\right.

$$\\

\hline

\end{tabular}

\caption{Transformation Rules}

\label{tab:trans}

$T\in \{\NEXT, \GLOBAL, \FUTURE\}$,

%$T'\in \{\UNTIL, \UNLESS\}$,

$ind$ is a new index and $Q\in \{\ALL, \EXIST\_{\tuple{ind}}\}$. Besides, $q$ is an atom, $l$ is a literal, $D$ is a disjunction of literals (possibly consisting of a single literal) and $\varphi$, $\varphi\_1$, and $\varphi\_2$ be \CTL\ formulae.

\end{table\*}

% \begin{align\*}

% & \textbf{Trans(1)}

% \end{align\*}

%The set $Trans$ of transformation rules~\cite{zhang2009refined} we will use is as follows:

% \begin{table\*}[]

% \centering

% %\footnotesize

% \begin{tabular}{l l l}

% %\specialrule{0em}{2pt}{2pt}

% \hline

% $\textbf{Trans(1)} \frac{\scriptstyle q \supset \EXIST T \varphi}{q\supset \EXIST\_{\tuple{ind}} T \varphi}$ & $\textbf{Trans(2)} \frac{q \supset \EXIST (\varphi\_1 T' \varphi\_2)}{q\supset \EXIST\_{\tuple{ind}} (\varphi\_1 T' \varphi\_2)}$ & $\textbf{Trans(3)} \frac{q\supset \varphi\_1 \wedge \varphi\_2}{\scriptsize \begin{array}{ll}

% q\supset \varphi\_1\\

% q\supset \varphi\_2

% \end{array} }$ \\

% $\textbf{Trans(4)} \frac{q\supset \varphi\_1 \vee \varphi\_2}{ \scriptsize \begin{array}{ll}

% q\supset \varphi\_1 \vee p,\ & \hbox{if $\varphi\_2$ is not}\\

% p\supset \varphi\_2,\ & \hbox{a disjunct}

% \end{array} }$ & $\textbf{Trans(5)}

% \begin{array}{ll}

% \frac{q\supset D}{\top \supset \neg q \vee D}\\

% \frac{q\supset \perp}{\top \supset \neg q}\\

% \frac{q \supset \top}{\{\}}

% \end{array} $

% & $\textbf{Trans(6)} \frac{q\supset Q\NEXT \varphi}{ \scriptsize \begin{array}{ll}

% q\supset Q\NEXT p,\ & \hbox{if $\varphi$ is not}\\

% p\supset \varphi,\ & \hbox{a disjunct}

% \end{array} }$ \\

% $\textbf{Trans(7)} \frac{q\supset Q\FUTURE \varphi}{\scriptsize \begin{array}{ll}

% q\supset Q\FUTURE p,\ & \hbox{if $\varphi$ is not}\\

% p\supset \varphi,\ & \hbox{a literal}

% \end{array} }$ &

% $\textbf{Trans(8)} \frac{q\supset Q(\varphi\_1 \UNTIL \varphi\_2)}{\scriptsize \begin{array}{ll}

% q\supset Q(\varphi\_1 \UNTIL p),\ & \hbox{if $\varphi\_2$ is not}\\

% p\supset \varphi\_2,\ & \hbox{a literal}

% \end{array} }$

% & $\textbf{Trans(9)} \frac{q\supset Q(\varphi\_1 \UNLESS \varphi\_2)}{\scriptsize \begin{array}{ll}

% q\supset Q(\varphi\_1 \UNLESS p),\ & \hbox{if $\varphi\_2$ is not}\\

% p\supset \varphi\_2,\ & \hbox{a literal}

% \end{array} }$ \\

% $\textbf{Trans(10)} \frac{q\supset Q\GLOBAL \varphi}{\scriptsize \begin{array}{ll}

% q \supset p\\

% p\supset \varphi\\

% p\supset Q\NEXT p

% \end{array} }$ &

% $\textbf{Trans(11)} \frac{q\supset Q(\varphi \UNTIL l)}{\scriptsize \begin{array}{ll}

% q \supset l\vee p\\

% p\supset \varphi\\

% p\supset Q\NEXT(l\vee p)\\

% q\supset Q \FUTURE l

% \end{array} }$

% & $\textbf{Trans(12)} \frac{q\supset Q(\varphi \UNLESS l)}{\scriptsize \begin{array}{ll}

% q \supset l\vee p\\

% p\supset \varphi\\

% p\supset Q\NEXT(l\vee p)

% \end{array} }$\\

% \hline

% \end{tabular}

% \caption{Transformation Rules}

% \label{tab:trans}

% Where $T\in \{\NEXT, \GLOBAL, \FUTURE\}$, $T'\in \{\UNTIL, \UNLESS\}$, $ind$ is a new index and $Q\in \{\ALL, \EXIST\_{\tuple{ind}}\}$. Besides, $q$ is an atom, $l$ is a literal, $D$ is a disjunction of literals (possible consisting of a single literal) and $\varphi$, $\varphi\_1$, and $\varphi\_2$ be \CTL\ formulae.

% \end{table\*}

\begin{table\*}[h]

\centering

\footnotesize

\begin{tabular}{l}

$$

\begin{aligned}

& \textbf{(SRES1)}\frac{P\supset \ALL\NEXT(C\vee l), Q\supset \ALL\NEXT(D\vee \neg l)}{P\wedge \ALL\NEXT(C\vee D)}; &&

\textbf{(SRES2)} \frac{P\supset \EXIST\_{\tuple{ind}} \NEXT(C\vee l), Q\supset \ALL\NEXT(D\vee \neg l)}{P\wedge Q \supset \EXIST\_{\tuple{ind}}\NEXT(C\vee D)};\\

& \textbf{(SRES3)} \frac{P\supset \EXIST\_{\tuple{ind}}\NEXT(C\vee l), Q \supset \EXIST\_{\tuple{ind}}\NEXT(D\vee \neg l)}{P\wedge Q\supset\EXIST\_{\tuple{ind}}\NEXT(C\vee D)}; &&

\textbf{(SRES4)} \frac{\start \supset C\vee l, \start \supset D \vee \neg l}{\start \supset C\vee D}; \\

& \textbf{(SRES5)} \frac{\top \supset C\vee l, \start \supset D \vee \neg l}{\start \supset C \vee D}; &&

\textbf{(SRES6)} \frac{\top \supset C \vee l, Q \supset \ALL\NEXT(D \vee \neg l)}{Q\supset \ALL \NEXT(C\vee D)}; \\

& \textbf{(SRES7)} \frac{\top \supset C \vee l, Q \supset \EXIST\_{\tuple{ind}} \NEXT(D \vee \neg l)}{Q\supset \EXIST\_{\tuple{ind}}\NEXT(C\vee D)}; &&

\textbf{(SRES8)} \frac{\top \supset C\vee l, \top \supset D \vee \neg l}{\top \supset C \vee D};\\

& \textbf{(RW1)} \frac{\bigwedge\_{i=1}^n m\_i \supset \ALL\NEXT \perp}{\top \supset \bigvee\_{i=1}^n \neg m}; && \textbf{(RW2)} \frac{\bigwedge\_{i=1}^n m\_i \supset \EXIST\_{\tuple{ind}}\NEXT \perp}{\top \supset \bigvee\_{i=1}^n \neg m}; \\

&\textbf{(ERES1)} \frac{\Lambda \supset \EXIST\_{\tuple{ind}} \NEXT \EXIST\_{\tuple{ind}}\GLOBAL l, Q \supset \ALL \FUTURE \neg l}{Q \supset \ALL(\neg \Lambda \UNLESS \neg l)}; && \textbf{(ERES2)} \frac{\Lambda \supset \EXIST\_{\tuple{ind}} \NEXT \EXIST\_{\tuple{ind}}\GLOBAL l, Q \supset \EXIST\_{\tuple{ind}} \FUTURE \neg l}{Q \supset \EXIST\_{\tuple{ind}}(\neg \Lambda \UNLESS \neg l)}.

\end{aligned}

$$\\

\end{tabular}

\caption{Resolution Rules}

$P$, $Q$ are conjunction of literals, $C$, $D$ are disjunction of literals and $l$ is a literal. Besides, $\Lambda=\bigvee\_{i=1}^n \bigwedge\_{i=1}^{m\_i}P\_j^i$ and $P\_j^i$ are conjunction of literals for all $1\leq i\leq n$ and $1\leq j\leq m$. Note that the resolvent of both $\textbf{(ERES1)}$ and $\textbf{(ERES2)}$ can be translated into a set of $\CTLsnf$ clauses (see~\cite{zhang2014resolution}). In this paper, we assume all of the results are $\CTLsnf$ clauses.

\label{tab:res}

\end{table\*}

% \begin{table\*}[h]

% \centering

% \begin{tabular}{l l}

% $\textbf{(SRES1)}\frac{P\supset \ALL\NEXT(C\vee l), Q\supset \ALL\NEXT(D\vee \neg l)}{P\wedge \ALL\NEXT(C\vee D)}$ & $\textbf{(SRES2)} \frac{P\supset \EXIST\_{\tuple{ind}} \NEXT(C\vee l), Q\supset \ALL\NEXT(D\vee \neg l)}{P\wedge Q \supset \EXIST\_{\tuple{ind}}\NEXT(C\vee D)}$ \\

% $\textbf{(SRES3)} \frac{P\supset \EXIST\_{\tuple{ind}}\NEXT(C\vee l), Q \supset \EXIST\_{\tuple{ind}}\NEXT(D\vee \neg l)}{P\wedge Q\supset\EXIST\_{\tuple{ind}}\NEXT(C\vee D)}$ & $\textbf{(SRES4)} \frac{\start \supset C\vee l, \start \supset D \vee \neg l}{\start \supset C\vee D}$ \\

% $\textbf{(SRES5)} \frac{\top \supset C\vee l, \start \supset D \vee \neg l}{\start \supset C \vee D}$ & $\textbf{(SRES6)} \frac{\top \supset C \vee l, Q \supset \ALL\NEXT(D \vee \neg l}{Q\supset \ALL \NEXT(C\vee D)}$ \\

% $\textbf{(SRES7)} \frac{\top \supset C \vee l, Q \supset \EXIST\_{\tuple{ind}} \NEXT(D \vee \neg l)}{Q\supset \EXIST\_{\tuple{ind}}\NEXT(C\vee D)}$ & $\textbf{(SRES8)} \frac{\top \supset C\vee l, \top \supset D \vee \neg l}{\top \supset C \vee D}$\\

% $\textbf{(RW1)} \frac{\bigwedge\_{i=1}^n m\_i \supset \ALL\NEXT \perp}{\top \supset \bigvee\_{i=1}^n \neg m}$ & $\textbf{(RW2)} \frac{\bigwedge\_{i=1}^n m\_i \supset \EXIST\_{\tuple{ind}}\NEXT \perp}{\top \supset \bigvee\_{i=1}^n \neg m}$ \\

% $\textbf{(ERES1)} \frac{\Lambda \supset \EXIST\_{\tuple{ind}} \NEXT \EXIST\_{\tuple{ind}}\GLOBAL l, Q \supset \ALL \FUTURE \neg l}{Q \supset \ALL(\neg \Lambda \UNLESS \neg l)}$ & $\textbf{(ERES2)} \frac{\Lambda \supset \EXIST\_{\tuple{ind}} \NEXT \EXIST\_{\tuple{ind}}\GLOBAL l, Q \supset \EXIST\_{\tuple{ind}} \FUTURE \neg l}{Q \supset \EXIST\_{\tuple{ind}}(\neg \Lambda \UNLESS \neg l)}$.

% \end{tabular}

% \caption{Resolution Rules}

% Where $P$, $Q$ are conjunction of literals, $C$, $D$ are disjunction of literals and $l$ is a literal. Besides, $\Lambda=\bigvee\_{i=1}^n \bigwedge\_{i=1}^{m\_i}P\_j^i$ and $P\_j^i$ are conjunction of literals for all $1\leq i\leq n$ and $1\leq j\leq m$. Note that the resolutions of both $\textbf{(ERES1)}$ and $\textbf{(ERES2)}$ can be translated into a set of $\CTLsnf$ clauses, see~\cite{zhang2009refined} for more detail. In this paper we assume all the results are the set of $\CTLsnf$ clauses.

% \label{tab:res}

% \end{table\*}

% \begin{align\*}

% & \textbf{(SRES1)}\frac{P\supset \ALL\NEXT(C\vee l), Q\supset \ALL\NEXT(D\vee \neg l)}{P\wedge \ALL\NEXT(C\vee D)}\\

% & \textbf{(SRES2)} \frac{P\supset \EXIST\_{\tuple{ind}} \NEXT(C\vee l), Q\supset \ALL\NEXT(D\vee \neg l)}{P\wedge Q \supset \EXIST\_{\tuple{ind}}\NEXT(C\vee D)}\\

% & \textbf{(SRES3)} \frac{P\supset \EXIST\_{\tuple{ind}}\NEXT(C\vee l), Q \supset \EXIST\_{\tuple{ind}}\NEXT(D\vee \neg l)}{P\wedge Q\supset\EXIST\_{\tuple{ind}}\NEXT(C\vee D)} \\

% & \textbf{(SRES4)} \frac{\start \supset C\vee l, \start \supset D \vee \neg l}{\start \supset C\vee D} \\

% & \textbf{(SRES5)} \frac{\top \supset C\vee l, \start \supset D \vee \neg l}{\start \supset C \vee D} \\

% & \textbf{(SRES6)} \frac{\top \supset C \vee l, Q \supset \ALL\NEXT(D \vee \neg l}{Q\supset \ALL \NEXT(C\vee D)}\\

% & \textbf{(SRES7)} \frac{\top \supset C \vee l, Q \supset \EXIST\_{\tuple{ind}} \NEXT(D \vee \neg l)}{Q\supset \EXIST\_{\tuple{ind}}\NEXT(C\vee D)} \\

% & \textbf{(SRES8)} \frac{\top \supset C\vee l, \top \supset D \vee \neg l}{\top \supset C \vee D}\\

% & \textbf{(RW1)} \frac{\bigwedge\_{i=1}^n m\_i \supset \ALL\NEXT \perp}{\top \supset \bigvee\_{i=1}^n \neg m} \\

% & \textbf{(RW2)} \frac{\bigwedge\_{i=1}^n m\_i \supset \EXIST\_{\tuple{ind}}\NEXT \perp}{\top \supset \bigvee\_{i=1}^n \neg m}\\

% % \end{align\*}

% % \begin{align\*}

% & \textbf{(ERES1)} \frac{\Lambda \supset \EXIST\_{\tuple{ind}} \NEXT \EXIST\_{\tuple{ind}}\GLOBAL l, Q \supset \ALL \FUTURE \neg l}{Q \supset \ALL(\neg \Lambda \UNLESS \neg l)} \\

% & \textbf{(ERES2)} \frac{\Lambda \supset \EXIST\_{\tuple{ind}} \NEXT \EXIST\_{\tuple{ind}}\GLOBAL l, Q \supset \EXIST\_{\tuple{ind}} \FUTURE \neg l}{Q \supset \EXIST\_{\tuple{ind}}(\neg \Lambda \UNLESS \neg l)}.

% \end{align\*}

% Where $P$, $Q$ are conjunction of literals, $C$, $D$ are disjunction of literals and $l$ is a literal. Besides, $\Lambda=\bigvee\_{i=1}^n \bigwedge\_{i=1}^{m\_i}P\_j^i$ and $P\_j^i$ are conjunction of literals for all $1\leq i\leq n$ and $1\leq j\leq m$.

For notational convenience, we use $V\subseteq \Ha$ to denote the set we want to forget, and $V' \subseteq \Ha$, with $V \cap V'={\O}$, to denote the set of atoms introduced in the computation and not in the given formula. Moreover, let $\varphi$ be the \CTL\ formula, $T\_{\varphi}$ be the set of $\CTLsnf$ clauses obtained from $\varphi$ by applying transformation rules on it and $\Hm=(S,R,L,[\\_], s\_0)$ (unless stated otherwise).

Moreover, let $T$ and $T'$ be formulae, $I$ a set of indexes introduced in the transformation and $V''\subseteq \Ha$. And by $T\equiv\_{\tuple{V'', I}} T'$ we mean that $\forall (\Hm, s\_0) \in \Mod(T)$, there is a $(\Hm', s\_0')$ s.t. $(\Hm,s\_0) \lrto\_{\tuple{V'', I}} (\Hm',s\_0')$ and $(\Hm', s\_0') \models T'$, and vice versa.

% \begin{definition}[\tuple{V,I}-Equation]\label{def:BisimEqu}

% Let $T$, $T'$ be two formulae (or sets of formulae), $I$ a set of indexes introduced in the transformation and $V\subseteq \Ha$. We say $T$ is \tuple{V,I}\-Equation, written $T\equiv\_{\tuple{V, I}} T'$, if

% \begin{enumerate}

% \item $\forall (\Hm, s\_0) \in \Mod(T)$ there is a $(\Hm', s\_0')$ such that $(\Hm,s\_0) \lrto\_{\tuple{V, I}} (\Hm',s\_0')$ and $(\Hm', s\_0') \models T'$, and

% \item $\forall (\Hm', s\_0') \in \Mod(T')$ there is a $(\Hm, s\_0)$ such that $(\Hm,s\_0) \lrto\_{\tuple{V, I}} (\Hm',s\_0')$ and $(\Hm, s\_0) \models T'$.

% \end{enumerate}

% % we mean that $\forall (\Hm, s\_0) \in \Mod(T)$ there is a $(\Hm', s\_0')$ such that $(\Hm,s\_0) \lrto\_{\tuple{V'', I}} (\Hm',s\_0')$ and $(\Hm', s\_0') \models T'$ and vice versa.

% \end{definition}

Algorithm~\ref{alg:compute:forgetting:by:Resolution} computes forgetting in \CTL.

The main idea is that, first we turn the given \CTL\ formula into a set of $\CTLsnf$ clauses (the \emph{transformation process}), and then we compute all the possible resolutions on the specified set of atoms (the \emph{resolution process}). Third, we eliminate all the irrelevant atoms in $V$ (the \emph{removal process}). % which does not be eliminated by the resolution.

% We will describe this process, which include \emph{Instantiate}, \emph{Connect} and \emph{Removing\\_atoms} sub-processes, in detail below.

As a final step, in order to change the obtained result into a \CTL\ formula, we need to go through two sub-processes: \emph{Removing\\_index} (removing the index from the formula) and $T\_\CTL$ (removing the $\start$ from $T$).

After introducing each process, in order to describe our algorithm clearly, we shall illustrate it over a running example. Altogether those running examples will amount to showing, given a \CTL\ formula $\varphi$, how to compute the $\CTLforget(\varphi, V)$ using our algorithm step by step.

\begin{algorithm}[!h]

\caption{Computing forgetting - A resolution-based method}% ??????

\label{alg:compute:forgetting:by:Resolution}

%\LinesNumbered %?????????

\KwIn{A CTL formula $\varphi$ and a set $V$ of atoms}% ????????

\KwOut{$\emph{ERes}(\varphi, V)$} %esult of forgetting $V$ from $\varphi$% ????

$T\_{\varphi}\lto {\O}$ // the initial set of $\CTLsnf$ clauses of $\varphi$ \;

%$T\_{\NI} \lto {\O}$ // the set of $\CTLsnf$ clauses without index\;

$V'\lto {\O}$ // the initial set of atoms introduced in the Transform process\;

$T\_{\varphi}, V' \lto \emph{Transform}(\varphi)$\;

$Res \lto \emph{Resolution}(T\_{\varphi}, V\cup V')$ \;

% $\Inst\_{V'} \lto \emph{Instantiate}(Res, V')$ \;

% $\Com\_{\EXIST\FUTURE} \lto \emph{Connect}(\Inst\_{V'})$ \;

$\emph{RemA} \lto \emph{Removing\\_atoms}(Res, V)$ \;

$\NI \lto \emph{Removing\\_index}(\emph{RemA})$ \; %Remove the index and start

%$\Rp \lto \emph{Replacing\\_atoms}(\NI)$\;

\Return $\bigwedge\_{\psi \in \NI\_{\CTL}} \psi$\;

\end{algorithm}

%\begin{figure}

% \centering

% % Requires \usepackage{graphicx}

% \includegraphics[width=7cm]{lct.png}\\

% \caption{The block diagram of the algorithm}\label{Fig:lct}

%\end{figure}

\subsection{Transformation Process}

The transformation process, represented as $\emph{Transform}(\varphi)$, transforms the \CTL\ formula into a set of $\CTLsnf$ clauses by using the transformation rules and returns a set $V'$ of atoms.

The transformation of any \CTL\ formula $\varphi$ into the set $T\_{\varphi}$ is a sequence $T\_0, T\_1,\dots, T\_n=T\_{\varphi}$ of sets of formulae with $T\_0=\{\ALL \GLOBAL(\start \supset p), \ALL \GLOBAL(p \supset \simp(\nnf(\varphi)))\}$~\footnote{The function $\nnf$ transforms a \CTL\ formula into a negation normal form (hence nnf), i.e., negative operations only occur before atoms, and $\simp$ is a function that uses the simplification rules in~\cite{zhang2014resolution} to simplify the formula.} such that for every $i$ (with $0 \leq i< n$), $T\_{i+1} = (T\_i \setminus \{\psi\}) \cup R\_i$~\cite{zhang2014resolution}) and all the formulae in $T\_{\varphi}$ are $\CTLsnf$ clauses where $p$ is a new atom (not occurring) in $\varphi$, $\psi$ is a formula in $T\_i$ which is not a $\CTLsnf$ clause, and $R\_i$ is the result set of applying a matching transformation rule to $\psi$. Note that throughout the transformation, formulae are kept in negation normal form (nnf).

%It has been shown in~\cite{zhang2009refined} that the transformation always exists for any \CTL\ formula. Moreover, t

% The following proposition shows that any \CTL\ formula $\varphi$ can be transformed into a set $T\_{\varphi}$ of $\CTLsnf$ clauses without effecting its satisfiability, and $\varphi$ is only different with $T\_{\varphi}$ on set $V'$ of atoms and set $I$ of indexes.

\begin{proposition}\label{pro:TranE}

Let $\varphi$ be a \CTL\ formula, then $\varphi \equiv\_{\tuple{V', I}} T\_{\varphi}$.

\end{proposition}

The above proposition shows that any \CTL\ formula $\varphi$ is only different with $T\_{\varphi}$ on set $V'$ of atoms and set $I$ of indexes. Now let us start with our running example.

% \begin{proof} (sketch)

% This can be proved from $T\_i$ to $T\_{i+1}$ $(0\leq i < n)$ by using one of the transformation rules on $T\_i$.

% We show $\varphi \equiv\_{\tuple{\{p\}, {\O}}} T\_0$. Other cases are similar.

% First, for every (\Hm\_1,s\_1) \in \Mod(\varphi)$, \ie $(\Hm\_1,s\_1) \models \varphi$. We can construct an initial \Ind-Kripke structure $\Hm\_2$ which is identical to $\Hm\_1$ except $L\_2(s\_2) = L\_1(s\_1) \cup \{p\}$. It is apparent that $(\Hm\_2,s\_2) \models T\_0$ and $(\Hm\_1, s\_1) \lrto\_{\tuple{\{p\}, {\O}}} (\Hm\_2, s\_2)$.

% Second, for all $(\Hm\_1,s\_1) \in \Mod(T\_0)$, it is apparent that $(\Hm\_1,s\_1) \models \varphi$ by the semantic of $\start$.

% %We will prove this proposition from the following several aspects:

% %

% % (1) $\varphi \equiv\_{\tuple{\{p\}, {\O}}} T\_0$.

% % $(\Rto)$ $\forall (\Hm\_1,s\_1) \in \Mod(\varphi)$, \ie $(\Hm\_1,s\_1) \models \varphi$. We can construct an \Ind-Kripke structure $\Hm\_2$ is identical to $\Hm\_1$ except $L\_2(s\_2) = L\_1(s\_1) \cup \{p\}$. It is apparent that $(\Hm\_2,s\_2) \models T\_0$ and $(\Hm\_1, s\_1) \lrto\_{\tuple{\{p\}, {\O}}} (\Hm\_2, s\_2)$.

% %

% % $(\Lto)$ $\forall (\Hm\_1,s\_1) \in \Mod(T\_0)$, it is apparent that $(\Hm\_1,s\_1) \models \varphi$ by the sematic of $\start$.

% %

% %By $\psi \rto\_t R\_i$ we mean using transformation rules $t$ on formula $\psi$ (the formulae $\psi$ as the

% %premises of rule $t$) and obtaining the set $R\_i$ of transformation results. Let $X$ be a set of formulas

% %we will show $T\_i \equiv\_{\tuple{V',I}} T\_{i+1}$ by using the transformation rule $t$. Where $T\_i= X \cup \{\psi\}$, $T\_{i+1}=X \cup R\_i$, $V'$ is the set of atoms introduced by $t$ and $I$ is the set of indexes introduced by $t$. (We will prove this result in $t\in \{$Trans(1), Trans(4), Trans(6)$\}$, other cases can be proved similarly.)

% %

% %(2) For $t$=Trans(1):\\

% % $(\Rto)$ $\forall (\Hm\_1,s\_1) \in \Mod(T\_i)$ \ie $(\Hm\_1, s\_1) \models X \wedge \ALL\GLOBAL(q \supset \EXIST \NEXT \varphi)$\\

% % $\Rto$ $(\Hm\_1,s\_1)\models X$ and for every $\pi$ starting from $s\_1$ and every state $s\_1^j \in \pi$, $(\Hm,s\_1^j) \models \neg q$ or there exists a path $\pi'$ starting from $s\_1^j$ such that there exists a state $s\_1^{j+1}$ such that $(s\_1^j,s\_1^{j+1})\in R\_1$ and $(\Hm,s\_1^{j+1})\models \varphi$\\

% % We can construct an \Ind-Kripke structure $\Hm\_2$ is identical to $\Hm\_1$ except $[ind]\_2= \bigcup\_{s\in S} R\_s \cup R\_y$, where $R\_{s\_1^{j}}=\{(s\_1^{j},s\_1^{j+1}), (s\_1^{j+1}, s\_1^{j+2}),\dots\}$ and $R\_y=\{(s\_x,s\_y)| \forall s\_x \in S$ if $\forall (s\_1',s\_2')\in \bigcup\_{s\in S} R\_s, s\_1'\neq s\_x$ then find a unique $s\_y\in S$ such that $(s\_x,s\_y)\in R\}$. It is apparent that $(\Hm\_1, s\_1) \lrto\_{\tuple{{\O}, \{ind\}}} (\Hm\_2, s\_2)$ (let $s\_2=s\_1$).\\

% % $\Rto$ for every path starting from $s\_1$ and every state $s\_1^j$ in this path, $(\Hm\_2, s\_1^j) \models \neg q$ or $(\Hm\_2, s\_1^j)\models \EXIST \NEXT \varphi\_{\tuple{ind}}$ \hfill (by the semantic of $\EXIST \NEXT$)\\

% % $\Rto$ $(\Hm\_2, s\_1) \models \ALL \GLOBAL(q \supset \EXIST\_{\tuple{ind}} \NEXT \varphi )$\\

% % $\Rto$ $(\Hm\_2, s\_1) \models X \wedge \ALL \GLOBAL(q \supset \EXIST\_{\tuple{ind}} \NEXT \varphi )$

% %

% % $(\Lto)$ $\forall (\Hm\_1,s\_1) \in \Mod(T\_{i+1})$ \ie $(\Hm\_1,s\_1) \models X \wedge \ALL \GLOBAL(q \supset \EXIST\_{\tuple{ind}} \NEXT \varphi )$\\

% % $\Rto$ $(\Hm\_1,s\_1) \models X$ and $(\Hm\_1,s\_1) \models \ALL \GLOBAL(q \supset \EXIST\_{\tuple{ind}} \NEXT \varphi)$\\

% % $\Rto$ for every path starting from $s\_1$ and every state $s\_1^j$ in this path, $(\Hm\_1, s\_1^j) \models \neg q$ or there exits a state $s'$ such that $(s\_1^j, s')\in [ind]\_1$ and $(\Hm\_1, s') \models \varphi$ \hfill (by the semantic of $\EXIST\_{\tuple{ind}} \NEXT$)\\

% % $\Rto$ for every path starting from $s\_1$ and every state $s\_1^j$ in this path, $(\Hm\_1, s\_1^j) \models \neg q$ or $(\Hm\_1, s\_1^j) \models \EXIST \NEXT \varphi$ \hfill (by the semantic of $\EXIST \NEXT$)\\

% % $\Rto$ $(\Hm\_1,s\_1) \models \ALL\GLOBAL(q \supset \EXIST \NEXT \varphi)$\\

% % $\Rto$ $(\Hm\_1, s\_1) \models X \wedge \ALL\GLOBAL(q \supset \EXIST \NEXT \varphi)$\\

% % It is apparent that $(\Hm\_1, s\_1) \lrto\_{\tuple{{\O}, \{ind\}}} (\Hm\_1, s\_1)$.

% %

% %(3) For $t$=Trans(4):\\

% % $(\Rto)$ $\forall (\Hm\_1,s\_1) \in \Mod(T\_i)$, \ie $(\Hm\_1,s\_1) \models X \wedge \ALL\GLOBAL (q \supset \varphi\_1 \vee \varphi\_2)$ \\

% % $\Rto$ $(\Hm\_1,s\_1) \models X$ and $\forall s\_1'\in S, (\Hm\_1,s\_1') \models q \supset \varphi\_1 \vee \varphi\_2$\\

% % $\Rto$ $(\Hm\_1,s\_1') \models \neg q$ or $(\Hm\_1,s\_1') \models \varphi\_1 \vee \varphi\_2$\\

% % The we can construct an \Ind-Kripke structure $\Hm\_2$ as follows. $\Hm\_2$ is the same with $\Hm\_1$ when $(\Hm\_1,s\_1') \models \neg q$. When $(\Hm\_1,s\_1') \models q$, $\Hm\_2$ is identical to $\Hm\_1$ except if $(\Hm\_1,s\_1') \models \varphi\_1$ then $L\_2(s\_1')= L\_1(s\_1')$ else $L\_2(s\_1') = L\_1(s\_1') \cup \{p\}$. It is apparent that $(\Hm\_2,s\_1') \models (q\supset \varphi\_1 \vee p) \wedge (p \supset \varphi\_2)$, then $(\Hm\_2,s\_1) \models T\_{i+1}$ and $(\Hm\_1, s\_1) \lrto\_{\tuple{\{p\}, {\O}}} (\Hm\_2, s\_2)$.

% %

% % $(\Lto)$ $\forall (\Hm\_1, s\_1) \in \Mod(T\_{i+1})$, \ie $(\Hm\_1,s\_1) \models X \wedge \ALL\GLOBAL (q\supset \varphi\_1 \vee p) \wedge \ALL\GLOBAL(p \supset \varphi\_2)$. It is apparent that $(\Hm\_1, s\_1) \models T\_i$.

% %

% %

% %(4) For $t$=Trans(6):\\

% %We prove for $\EXIST\_{\tuple{ind}} \NEXT$, while for the $\ALL \NEXT$ can be proved similarly.

% %

% % $(\Rto)$ $\forall (\Hm\_1,s\_1) \in \Mod(T\_i)$, \ie $(\Hm\_1,s\_1) \models X \wedge \ALL\GLOBAL(q \supset \EXIST\_{\tuple{ind}}\NEXT \varphi)$\\

% % $\Rto$ $(\Hm\_1,s\_1) \models X$ and $\forall s\_1'\in S, (\Hm\_1,s\_1') \models q \supset \EXIST\_{\tuple{ind}} \NEXT \varphi$\\

% % $\Rto$ $(\Hm\_1,s\_1') \models \neg q$ or there exists a state $s'$ such that $(s\_1', s') \in [ind]$ and $(\Hm\_1,s') \models \varphi$ \\

% % We can construct an \Ind-Kripke structure $\Hm\_2$ as follows. $\Hm\_2$ is the same with $\Hm\_1$ when $(\Hm\_1,s\_1') \models \neg q$. When $(\Hm\_1,s\_1') \models q$, $\Hm\_2$ is identical to $\Hm\_1$ except for $s'$ there is $L\_2(s') = L\_1(s') \cup \{p\}$. It is apparent that $(\Hm\_2,s\_1) \models \ALL\GLOBAL(q\supset \EXIST\_{\tuple{ind}} \NEXT p) \wedge \ALL\GLOBAL(p \supset \varphi)$, $(\Hm\_2,s\_2) \models T\_{i+1}$ and $(\Hm\_1, s\_1) \lrto\_{\tuple{\{p\}, {\O}}} (\Hm\_2, s\_2)$ ($s\_2=s\_1$).

% %

% % $(\Lto)$ $\forall (\Hm\_1, s\_1) \in \Mod(T\_{i+1})$, \ie $(\Hm\_1,s\_1) \models X \wedge \ALL\GLOBAL(q\supset \EXIST\_{\tuple{ind}} \NEXT p) \wedge \ALL\GLOBAL(p \supset \varphi)$. It is apparent that $(\Hm\_1, s\_1) \models T\_i$.

% \end{proof}

%This means that models of $\varphi$ and $T\_{\varphi}$ are $\tuple{V', I}$-bisimular.

%, i.e. except that the label function for those atoms in $V'$ and the relations $[i]$ with $i\in I$ may be different in those models.

% \begin{algorithm}[!h]

% \caption{$\emph{Transform}(\varphi)$}% ??????

% \label{alg:compute:transformation}

% %\LinesNumbered %?????????

% \KwIn{A CTL formula $\varphi$}% ????????

% \KwOut{A set $T\_{\varphi}$ of $\CTLsnf$ clauses and a set $V'$ of atoms}% ????

% $T\_{\varphi}\lto {\O}$ // the initial set of $\CTLsnf$ clauses of $\varphi$ \;

% $OldT\lto \{\start \supset z, z \supset \simp(\nnf(\varphi))\}$\;

% $V'\lto \{z\}$\;

% \While {$true$} {

% $R\lto {\O}$\;

% $X\lto {\O}$\;

% \If {Chose a formula $\psi\in OldT$ that does not a $\CTLsnf$ clause}{

% Using a match rule $Rl$ to transform $\psi$ into a set $R$ of $\CTLsnf$ clauses\;

% $X$ is the set of atoms introduced by using $Rl$\;

% $V' \lto V' \cup X$\;

% $T\_{\varphi}\lto OldT\setminus \{\psi\} \cup R$\;

% }

% \Else {\bf break\;}

% $OldT\lto T\_{\varphi}$\;

% }

% \Return $T\_{\varphi}$, $V'$\;

% \end{algorithm}

\begin{example}\label{examp:Tran}[Running example]

Let $\varphi=\ALL((p\wedge q) \UNTIL (f\vee m)) \wedge r$ and $V=\{p,r\}$.

As the first step, \emph{Transform}$(\varphi)$ will yield the result $T\_{\varphi}$ which can be listed as follows:

\begin{align\*}

& 1. \start\supset z && 2. \top \supset \neg z \vee r && 3.\top \supset \neg x\vee f \vee m\\

& 4. \top \supset \neg z \vee x \vee y && 5.\top \supset \neg y \vee p && 6.\top \supset \neg y \vee q\\

& 7. z \supset \ALL \FUTURE x && 8. y \supset \ALL \NEXT(x\vee y).

\end{align\*}

Besides, the set of new atoms introduced in this process is $V'=\{x, y,z, w\}$ in which $w$ is a new atom related to $z \supset \ALL \FUTURE x$.~\footnote{Note that we always generate a new atom for each $T$-sometime clause with $T\in \{\EXIST, \ALL\}$ when this clause is produced during the transform process.} %~\cite{zhang2014resolution}.

\end{example}

\subsection{Resolution Process}

The resolution process consists of computing all the possible \emph{resolutions} of $T\_{\varphi}$ on $V\cup V'$, represented by the function $\emph{Resolution}(T\_{\varphi}, V\cup V')$.

Let $C$ and $C'$ be two formulae, we say $C$ and $C'$ are resolvable if there is a resolution rule using $C$ and $C'$ as the premises on some given atom.

% In this way, if $C$ and $C'$ are resolvable, then $res(C,C')$ is a set of $\CTLsnf$ clauses obtained by using the matching resolution rule on $C$, $C'$ and the given atom.

Moreover, a \emph{derivation} on a set $V\cup V'$ of atoms and $T\_{\varphi}$ is a sequence $T\_0=T\_{\varphi}, T\_1, T\_2$, $\dots$, $T\_n=Res$ of sets of $\CTLsnf$ clauses such that $T\_{i+1} = T\_i \cup R\_i$ for all $0\leq i < n$ and no two formulae in $Res$ are resolvable, where $R\_i$ is a set of clauses obtained as the resolvent of the application of a resolution rule to premises in $T\_i$.

Note that all $T\_i$ (where $0 \leq i \leq n$) are set of $\CTLsnf$ clauses.

% It has also been shown in~\cite{zhang2014resolution} that the derivation on any set of atoms always exists for any given set of $\CTLsnf$ clauses.

And if there is a $T\_i$ containing $\start\supset \perp$ or $\top\supset \perp$, then we can easily check that $\CTLforget(\varphi, V)\equiv\perp$.

%The pseudocode of the \emph{Resolution} process is shown in Algorithm~\ref{alg:compute:Res}.

% Let $C$ be a clause and $C'$ be a clause or set of clauses. If $C$ and $C'$ are resolvable, then $res(C,C')$ is a set of $\CTLsnf$ clauses, i.e., if there is a resolution rule using $C$ and $C'$ as the premises on some given atom.

\begin{proposition}\label{pro:ResE}

Let $\varphi$ be a \CTL\ formula,

%and $W$ be the set of new atoms introduced by resolution rules \textbf{(ERES1)} and \textbf{(ERES2)} (if any),

then $T\_{\varphi} \equiv\_{\tuple{V \cup V', {\O}}} Res$.

\end{proposition}

% \begin{proof}(sketch)

% This can be proved from $T\_i$ to $T\_{i+1}$ $(0\leq i < n)$ by using one resolution rule on $T\_i$.

% For instance, if we can use the resolution rule (SRES1) on $\psi\subseteq T\_i$ and obtain the result $R$, then we can prove $T\_i \equiv T\_{i+1}$ with $T\_{i+1} = T\_i \cup R$ as follows.

% On the one hand, it is apparent that $\psi \models R$ and then $T\_i \models T\_{i+1}$. On the other hand, $T\_i\subseteq T\_{i+1}$ and then $T\_{i+1} \models T\_i$.

% % By $\psi \rto\_r R\_i$ we mean using resolution rules $r$ on set $\psi$ (the formulae in $\psi$ as the premises of rule $r$) and obtaining the set $R\_i$ of resolution results.

% % we will show $T\_i \equiv\_{\tuple{V,I}} T\_{i+1}$ by using the resolution rule $r$. Where $T\_i= X \cup \psi$, $T\_{i+1}=X \cup R\_i$, $X$ be a set of $\CTLsnf$ clauses, $p$ be the proposition corresponding with literal $l$ used to do resolution in $r$.

% % (1) If $\psi \rto\_r R\_i$ by an application of $r\in \{\textbf{(SRES1)}, \dots, \textbf{(SRES8)}, \textbf{RW1}, \textbf{RW2}\}$, then $T\_i \equiv\_{\tuple{\{p\}, {\O}}} T\_{i+1}$.

% % On one hand, it is apparent that $\psi \models R\_i$ and then $T\_i \models T\_{i+1}$. On the other hand, $T\_i\subseteq T\_{i+1}$ and then $T\_{i+1} \models T\_i$.

% %

% % (2) If $\psi \rto\_r R\_i$ by an application of $r=$\textbf{(ERES1)},

% % then $T\_i \equiv\_{\tuple{\{l, w\_{\neg l}^{\ALL}\}, {\O}}} T\_{i+1}$.

% % It has been proved that $\psi \models R\_i$ in~\cite{bolotov2000clausal}, then there is $T\_{i+1}=T\_i \cup \Lambda\_{\neg l}^{\ALL}$ and then $\forall (\Hm\_1,s\_1) \in \Mod(T\_i= X \cup \psi)$ there is a $(\Hm\_2, s\_2)\in \Mod(T\_{i+1}=T\_i \cup \Lambda\_{\neg l}^{\ALL})$ s.t. $(\Hm\_1, s\_1) \lrto\_{\tuple{\{p, w\_{\neg l}^{\ALL}\}, {\O}}} (\Hm\_2, s\_2)$ and vice versa by Proposition~\ref{pro:TranE}.

% %

% %For rule \textbf{(ERES2)} we have the same result.

% \end{proof}

Proposition~\ref{pro:ResE} together with Proposition~\ref{pro:TranE} imply that $\varphi \equiv\_{\tuple{V \cup V', I}} Res$. That is, for any formula $\psi$ if $\IR(\psi, V\cup V')$, then $\varphi \models \psi$ iff $Res \models \psi$.

%, this resolves a part of the problem (1), i.e. connect the \CTL\ formula with $\CTLsnf$ formula.

More specifically, it means that $\varphi$ is different than $Res$ on $V \cup V'$ and $I'$. This gives us a guidance on which atoms and indices to eliminate (in $V \cup V'$ and in $I'$, respectively) in order to compute forgetting $V$ from $\varphi$.

%In the following subsections we will show how to do this, and before that let's see the resolvent of Example~\ref{examp:Res}. % by using Algorithm~\ref{alg:compute:Res}.

% \begin{algorithm}[!h]

% \caption{$\emph{Resolution}(T,V \cup V')$}% ??????

% \label{alg:compute:Res}

% %\LinesNumbered %?????????

% \KwIn{A set $T\_{\varphi}$ of $\CTLsnf$ clauses and a set $V\cup V'$ of atoms}% ????????

% \KwOut{A set $Res$ of $\CTLsnf$ clauses}% ????

% $S\lto \{C | C\in T\_{\varphi}$ and $\Var(C) \cap (V\cup V')= {\O}\}$\;

% $\Pi\lto T\setminus S$ \;

% \For {($p\in V\cup V')$} {

% $\Pi'\lto \{C \in \Pi| p\in \Var(C)\}$ \;

% $\Sigma \lto \Pi \setminus \Pi'$\;

% \For {($C\in \Pi'$ s.t. $p$ appearing in $C$ positively)} {

% \For {($C'\in\Pi'$ s.t. $p$ appearing in $C'$ negatively and $C$, $C'$ are resolvable)}{

% $\Sigma \lto \Sigma \cup res(C,C')$\;

% $\Pi' \lto \Pi' \cup \{C''\in res(C,C') | p\in \Var(C'')\}$\;

% }

% }

% $\Pi\lto \Sigma$\;

% }

% $Res\lto \Pi \cup S$\;

% \Return $Res$\;

% \end{algorithm}

\begin{example}\label{examp:Res}[cont'd from Example 1]

The resolvent of $T\_{\varphi}$ obtained from Example~\ref{examp:Tran} on $V\cup V'$ are the following clauses (in addition to the ones in Example~\ref{examp:Tran}):

\begin{align\*}

&(1)\ \start \supset r && (1,2,SRES 5)\\

&(2)\ \start \supset x \vee y && (1,4,SRES 5)

\end{align\*}

\begin{align\*}

&(3)\ \top \supset \neg z \vee y \vee f \vee m && (3, 4, SRES 8)\\

&(4)\ y \supset \ALL\NEXT(f\vee m\vee y) && (3,8, SRES 6)\\

&(5)\ \top \supset \neg z \vee x \vee p && (4,5, SRES 8)\\

&(6)\ \top \supset \neg z \vee x \vee q && (4,6, SRES 8)\\

&(7)\ y \supset \ALL\NEXT(x\vee p) && (5, 8, SRES 6)\\

&(8)\ y \supset \ALL\NEXT(x\vee q) && (6, 8, SRES 6)\\

&(9)\ \start \supset f\vee m \vee y && (3,(2), SRES 5) \\

% \end{align\*}

% \begin{align\*}

&(10)\ \start \supset x \vee p && (5,(2),SRES 5) \\

&(11)\ \start \supset x \vee q && (6,(2), SRES 5)\\

& (12)\ \top \supset p \vee \neg z \vee f \vee m && (5,(3), SRES 8)\\

& (13)\ \top \supset q \vee \neg z \vee f \vee m && (6,(3), SRES 8)\\

&(14)\ y \supset \ALL\NEXT(p \vee f\vee m) && (5, (4), SRES 6)\\

&(15)\ y \supset \ALL\NEXT(q \vee f\vee m) && (6, (4), SRES 6)\\

%\end{align\*}

%\begin{align\*}

&(16)\ \start \supset f\vee m \vee p && (5, (9), SRES 5) \\

&(17)\ \start \supset f\vee m \vee q && (6, (9), SRES 5)

\end{align\*}

\end{example}

\subsection{Elimination Process}

Given a formula (assuming all the implications ($\supset$) are written in the form of disjunctions) ($\vee$), we say that an atom occurs in the formula positively if it is preceded by an even number of negative connectives, else it occurs negatively. Moreover, we say that a formula $\varphi$ is positive w.r.t. $p$ if every $p$ occurs in $\varphi$ positively. Similarly, a formula $\varphi$ is negative w.r.t. $p$ if every $p$ occurs in $\varphi$ negatively.

For solving the aforementioned problem of \emph{how to eliminate irrelevant atoms}, we should focus on the following properties that are obtained from the transformation and resolution rules:

\begin{itemize}

\item \textbf{(GNA)} For all $p \in \Var(\varphi)$, $p$ does not occur positively on the left hand side of the $\CTLsnf$ clause;

%\item \textbf{(CNI)} for each global clause, there must be an atom $p\in V'$ appearing in the right hand negatively;

\item \textbf{(PI)} For all $p\in V'$, if $p$ occurs on the left hand side of a $\CTLsnf$ clause, then $p$ occurs positively.

\end{itemize}

% The \emph{Elimination} process includes two sub-processes \emph{Removing\\_index}. We describe them in the follows.

For eliminating those irrelevant atoms in $V$, we define the following \emph{Removing\\_atoms} operator.

\begin{definition}[Removing\\_atoms]\label{def:Elm}

%\textbf{(Elimination)}

Let $C$ be a $\CTLsnf$ clause and $V$ be a set of atoms, then the \emph{Removing\\_atoms} operator is defined as:

$$ \emph{Removing\\_atoms}(C, V)=\left\{

\begin{aligned}

\top, && \text{if}\ \Var(C) \cap V \neq \emptyset \\

C, && else.

\end{aligned}

\right.

$$

\end{definition}

Intuitively, if $C$ contains at least one of the atoms in $V$, then Removing\\_atoms$(C, V)$ is true, else it is $C$ itself.

For any set $T$ of formulae, we define that Removing\\_atoms$(T, V) = \{$Removing\\_atoms$(r, V) | r \in T\}$.

\begin{proposition}\label{pro:remove}

Let $V''=V \cup V'$, then we have

\[

Res \equiv\_{V''} \emph{Removing\\_atoms}(Res, V).

\]

\end{proposition}

Above proposition says that Removing\\_atoms does not affect the consequences of $Res$ on $\Ha - V''$, i.e., for each formula $\varphi$, if $\IR(\varphi, V'')$, then $Res \models \varphi$ iff Removing\\_atoms$(Res, V)\models \varphi$.

% \begin{proof}

% For convenience, we let $V=\{p\}$, i.e. $V$ contain only one element $p$, $C\_i$ is a classical clause and $l$ is $p$ or $\neg p$.

% It is evident that $Res \models \emph{Removing\\_atoms}(Res, V)$, hence we only need to prove that for each ${\cal K}=(\Hm, s)\in \Mod(\emph{Removing\\_atoms}(Res, V))$ with $\Hm=(S, R, L, s)$ there is an initial structure ${\cal K}'=(\Hm', s')$ such that ${\cal K} \lrto\_{V''} {\cal K}'$ and ${\cal K}' \models Res$.

% As we can see that the $p$ can only appear in the right of a clause, we will prove this proposition from the following several points.

% (1) We consider there are global clauses in $Res$ (the other cases are sub-cases of this one), then for each $C=\top\supset C\_1 \vee l \in Res$:

% (a) If there does not exist a clause $C'\in Res$ such that $C$ and $C'$ are resolvable on $p$, this means there is no other clauses in $Res$ except $Pt$-sometime clauses $C'$ containing $\neg l$ with $Pt\in \{\ALL, \EXIST\}$.

% If $p\not \in \Var(C')$, for each ${\cal K}=(\Hm, s)\in \Mod(\emph{Removing\\_atoms}(Res, V))$ we can construct $(\Hm',s')$ as follows: Let $\Hm'= (S, R, L',s)$ (i.e. $s'=s$) in which $L'$ is the same as $L$ except for each $s\_1\in S$, if $(\Hm, s\_1) \not \models C\_1 \vee l$ then let $L'(s\_1) = L(s\_1) \cup \{p\}$ if $l=p$ else $L'(s\_1) = L(s\_1) - \{p\}$.

% If $C'= Q\supset Pt \FUTURE \neg l$, without loss of generality, we assume $l=p$ for each ${\cal K}=(\Hm, s)\in \Mod(\emph{Removing\\_atoms}(Res, V))$ we construct $(\Hm',s')$ as follows: let $\Hm'=(S', R', L', s')$ with $S'=S$, $R'=R$, $s'=s$ and $L'=L$ except that for each $s\in S'$ we have $L'(s) = L(s) - \{Q\}$ if $Q$ is an atom (if $Q$ is a term then we can delete the atoms which appearing in $Q$ positively and add the atoms which appearing in $Q$ negatively) and $L'(s) = L(s) \cup \{p\}$ if $(\Hm, s) \not \models C\_1$ else $L'(s) = L(s)$.

% It is easy to check that ${\cal K} \lrto\_{V''} {\cal K}'$ and ${\cal K}' \models Res$.

% (b) If there are some clauses $C'\in Res$ such that $C$ and $C'$ are resolvable on $p$:

% \begin{enumerate}[(i)]

% \item If $C'= Q\supset Pt \NEXT (C\_2 \vee \neg l)$ (we let $Pt=\GLOBAL$, we can prove similarly for $Pt = \EXIST$) then we have $Q\supset \GLOBAL \NEXT(C\_1 \vee C\_2) \in Res$, then for each ${\cal K}=(\Hm, s)\in \Mod(\emph{Removing\\_atoms}(Res, V))$ we construct $(\Hm',s')$ as follows: Let $\Hm'= (S, R, L',s)$ (i.e. $s'=s$) in which $L'$ is the same as $L$ except for each $s\_1\in S$ if $(\Hm, s\_1) \not \models Q$ then for each $(s\_1, s\_2) \in R$ if $(\Hm, s\_2) \not \models C\_1$ then let $L'(s\_2) = L(s\_2) \cup \{p\}$ if $l=p$ else $L'(s\_2) = L(s\_2) - \{p\}$, else if $(\Hm, s\_2) \models C\_1 \wedge \neg C\_2$ then let $L'(s\_2) = L(s\_2) - \{p\}$ if $l=p$ else $L'(s\_2) = L(s\_2) \cup \{p\}$; else if $(\Hm, s\_2) \models \neg C\_1 \wedge C\_2$ then let $L'(s\_2) = L(s\_2) \cup \{p\}$ if $l=p$ else $L'(s\_2) = L(s\_2) - \{p\}$. It is easy to check that ${\cal K} \lrto\_{V''} {\cal K}'$ and ${\cal K}' \models C' \wedge C$.

% \item If $C' = Q\supset Pt \FUTURE \neg l$. Without loss of generality, we assume $l=p$ for convenience. In order to make $C$ and $C'$ are resolvable on $p$, there must be a set of $\CTLsnf$ clauses $\{P\_1^1 \supset \* C\_1^1$, \dots, $P\_{m\_1}^1 \supset \* C\_{m\_1}^1$, $P\_1^n \supset \* C\_1^n$, \dots, $P\_{m\_n}^1 \supset \* C\_{m\_n}^1 \}$ such that $\*$ is either empty or

% an operator in $\{\GLOBAL \NEXT, \EXIST\_{\tuple{ind}} \NEXT\}$, which include $\neg C\_1 \supset l$, such that $\bigvee\_{i=1}^n \bigwedge\_{j=1}^{m\_i} P\_j^i \supset \EXIST \NEXT \EXIST \GLOBAL l$. Therefore, we get a clause $C''=\top \supset \neg Q \vee \neg p \vee C\_1$ by using ERES1 (similar for ERES2) and then $\top \supset \neg Q \vee C\_1$ by using SRES8 on $C$ and $C''$. In this case, for any ${\cal K}=(\Hm, s)\in \Mod(\emph{Removing\\_atoms}(Res, V))$ we construct $(\Hm',s')$ as follows: Let $\Hm'= (S, R, L',s)$ (i.e. $s'=s$) in which $L'$ is the same as $L$ except for each $s\_1\in S$ if $(\Hm, s\_1) \models Q$ then let $L'(s\_1) = L(s\_1) - \{p\}$, else $L'(s\_1) = L(s\_1) \cup \{p\}$. It is easy to check that ${\cal K} \lrto\_{V''} {\cal K}'$ and ${\cal K}' \models C' \wedge C$.

% \item We can consider other clauses similarly, and obtained that ${\cal K} \lrto\_{V''} {\cal K}'$ and ${\cal K}' \models Res$.

% \end{enumerate}

% (2) We consider the $Pt$-step clauses, let $C\in Res$ is $Q \supset \GLOBAL \NEXT(C\_1 \vee \neg l)$. Without loss of generality, we assume there are some clauses $C'\in Res$ such that $C$ and $C'$ are resolvable on $p$ and $l=p$.

% If $C'= Q\_1\supset Pt \NEXT (C\_2 \vee \neg l)$ (we let $Pt=\EXIST\_{ind}$, we can prove similarly for $Pt = \GLOBAL$) then we have $Q \wedge Q\_1 \supset \EXIST\_{ind} \NEXT(C\_1 \vee C\_2) \in Res$, then for each ${\cal K}=(\Hm, s)\in \Mod(\emph{Removing\\_atoms}(Res, V))$ we construct $(\Hm',s')$ as follows: Let $\Hm'= (S, R, L',s)$ (i.e. $s'=s$) in which $L'$ is the same as $L$ except for each $s\_1\in S$

% \begin{enumerate}[(i)]

% \item if $(\Hm, s\_1) \not \models Q \wedge Q\_1$ then ``if $(\Hm, s\_1) \models \neg Q \wedge Q\_1$ then (if $(\Hm, s\_2') \not \models C\_2$ for $(s\_1, s\_2') \in \pi\_s^{\tuple{ind}}$ then let $L'(s\_2') = L(s\_2') - \{p\}$ else $L'(s\_2') = L(s\_2')$), else if $(\Hm, s\_1) \models Q \wedge \neg Q\_1$ then for each $(s\_1, s\_2) \in R$ (if $(\Hm, s\_2) \not \models C\_1$ then let $L'(s\_2) = L(s\_2) \cup \{p\}$ else $L'(s\_2') = L(s\_2')$), else let $L'(s\_2') = L(s\_2')$".

% \item else if $(\Hm, s\_1) \models Q \wedge Q\_1$ then we have $(\Hm,s\_2') \models C\_1 \vee C\_2$ for $(s\_1, s\_2) \in \pi\_s^{\tuple{ind}}$. Therefore, if $(\Hm, s\_2') \models C\_1 \wedge \neg C\_2$ then $L'(s\_2') = L(s\_2') - \{p\}$, else if $(\Hm, s\_2') \models \neg C\_1 \wedge C\_2$ then let $L'(s\_2) = L(s\_2) \cup \{p\}$ else $L'(s\_2') = L(s\_2')$. For other state $s\_2$ with $(s\_1, s\_2) \in R$ and $s\_2 \not = s\_2'$, if $(\Hm, s\_1) \models Q$ and $(\Hm, s\_2) \models \neg C\_1$ then let $L'(s\_2) = L(s\_2) \cup \{p\}$ else $L'(s\_2') = L(s\_2')$.

% \end{enumerate}

% It is easy to check that ${\cal K} \lrto\_{V''} {\cal K}'$ and ${\cal K}' \models C' \wedge C$, in which ${\cal K}' = (\Hm',s')$.

% \end{proof}

\begin{example}\label{examp:remA}[cont'd from Example 2]

After removing the clauses that include atoms in $V=\{p\}$, the following clauses are left:

\begin{align\*}

& \start\supset z && \top \supset q \vee \neg z \vee f \vee m \\

& \top \supset \neg x\vee f \vee m && \top \supset \neg z \vee x \vee y \\

& \start \supset f\vee m \vee q && \top \supset \neg y \vee q\\

& z \supset \ALL \FUTURE x && y \supset \ALL \NEXT(x\vee y) \\

& y \supset \ALL\NEXT(q \vee f\vee m) && \start \supset x \vee y \\

& \top \supset \neg z \vee y \vee f \vee m && y \supset \ALL\NEXT(f\vee m\vee y)\\

& \top \supset \neg z \vee x \vee q && y \supset \ALL\NEXT(x\vee q) \\

& \start \supset f\vee m \vee y && \start \supset x \vee q

%& \start \supset f\vee m \vee q

\end{align\*}

\end{example}

% In this case, if we do not specify $l$, $C\_2$, $C\_3$ and $C\_4$ are instantiate formulae of $\Sub(Res, V')$, it is easy to check that all results including $P\supset \EXIST\_{\tuple{ind}}\NEXT (\neg l \vee C\_2 \vee C\_4)$ and $P\supset \ALL \NEXT (\neg l \vee C\_2 \vee C\_4)$ obtained from the \emph{Connect} process will be deleted in the Removing\\_atoms process.

\subsection{Removal of Indexes and Start}

%The $\emph{Removing\\_index}(\Gamma)$ process is to change the set $\Gamma$ of formulas into a set of formulas without the index by using the equations in Proposition~\ref{pro:In2NI}.

%The following proposition is important for our algorithm since ... .

Next, we explain the idea on how to remove the index and \textbf{start} from the formulas. First, the following proposition is crucial in eliminating the indexes in the set of clauses.

\begin{proposition}\label{pro:Ind:EF}

Let $\EXIST\_{\tuple{ind}} \FUTURE \varphi$ be a $\CTLsnf$ formula, then we have

\[

\EXIST\_{\tuple{ind}} \FUTURE \varphi \equiv \varphi \vee \EXIST\_{\tuple{ind}} \NEXT \EXIST\_{\tuple{ind}}\FUTURE \varphi.

\]

\end{proposition}

% \begin{proof}

% ($\Rto$) Let $(\Hm, s\_0) \in \Mod(\EXIST\_{\tuple{ind}} \FUTURE \varphi)$, then there exists a path $\pi\_{s\_0}^{\tuple{ind}}$ such that $(\Hm, s\_j) \models \varphi$ for some $s\_j \in \pi\_s^{\tuple{ind}}$ with $0 \leq j$. In this case, we can see either $j=0$ or $j > 0$, then we have $(\Hm, s\_0) \models \varphi \vee \EXIST\_{\tuple{ind}} \NEXT \EXIST\_{\tuple{ind}}\FUTURE \varphi$.

% ($\Lto$) Let $(\Hm, s\_0) \in \Mod(\varphi \vee \EXIST\_{\tuple{ind}} \NEXT \EXIST\_{\tuple{ind}}\FUTURE \varphi)$, then we have $(\Hm,s\_0) \models \varphi$ or there exists a path $\pi\_{s\_0}^{\tuple{ind}} = (s\_0, s\_1, \dots)$ such that $(\Hm, s\_1) \models \EXIST\_{\tuple{ind}}\FUTURE \varphi$. Therefore, we have $(\Hm, s\_0) \models \EXIST\_{\tuple{ind}} \FUTURE \varphi$ by the semantic of $\EXIST\_{\tuple{ind}}\FUTURE$.

% \end{proof}

By the transformation rules in Table~\ref{tab:trans} we know that there are no two $\CTLsnf$ clauses, $\EXIST$-step and $\EXIST$-sometime clauses, which have the same index during the transform process. Moreover, no two $\EXIST$-sometime clauses have the same index, and the Resolution process will not produce any $\EXIST$-sometime clauses.

Therefore, we can use Proposition~\ref{pro:Ind:EF} to transform a $\EXIST$-sometime clause into a similar $\EXIST$-step clause at first, and then use the following proposition to eliminate the indices.

%The following proposition is important for our algorithm since ... .

\begin{proposition}\label{pro:In2NI}

Let $P$, $P\_i$ and $\varphi\_i$ be \CTL\ formulas, then

\begin{enumerate}[(i)]

\item $\bigwedge\_{i=1}^n (P\supset \EXIST\_{\tuple{ind}} \NEXT \varphi\_i) \equiv\_{\tuple{\emptyset, \{ind\}}} P\supset \EXIST \NEXT \bigwedge\_{i=1}^n \varphi\_i$,

\item $\bigwedge\_{i=1}^n (P\_i\supset \EXIST\_{\tuple{ind}} \NEXT \varphi\_i) \equiv\_{\tuple{\emptyset, \{ind\}}} \bigwedge\_{e \in 2^{\{1,\dots, n\}} \setminus \{\emptyset\}}(\bigwedge\_{i\in e}P\_i\supset \EXIST \NEXT (\bigwedge\_{i\in e}\varphi\_i))$,

\item $\bigwedge\_{i=1}^n (P\supset \EXIST\_{\tuple{ind}} \FUTURE \varphi\_i) \equiv\_{\tuple{\emptyset, \{ind\}}} P\supset \bigvee\EXIST\FUTURE (\varphi\_{j\_1} \wedge \EXIST\FUTURE(\varphi\_{j\_2} \wedge \EXIST\FUTURE(\dots \wedge \EXIST\FUTURE \varphi\_{j\_n})))$, where $(j\_1, \dots, j\_n)$ is a sequence of all elements in $\{1, \dots, n\}$,

\item $P\supset (C \vee \EXIST\_{\tuple{ind}} \NEXT \varphi\_1) \wedge P \supset \EXIST\_{\tuple{ind}} \NEXT \varphi\_2 \equiv\_{\tuple{\emptyset, \{ind\}}} P \supset ((C \wedge \EXIST \NEXT \varphi\_2) \vee \EXIST \NEXT (\varphi\_1 \wedge \varphi\_2))$,

\item $(P\supset (C \vee \EXIST\_{\tuple{ind}} \NEXT \varphi\_1)) \vee (P \supset \EXIST\_{\tuple{ind}} \NEXT \varphi\_2) \equiv\_{\tuple{\emptyset, \{ind\}}} P \supset (C \vee \EXIST \NEXT (\varphi\_1 \vee \varphi\_2))$.

\end{enumerate}

\end{proposition}

% \begin{proof}

% (i) For all $(\Hm, s\_0) \in \Mod(\bigwedge\_{i=1}^n (P\supset \EXIST\_{\tuple{ind}} \NEXT \varphi\_i))$ there exists $(s\_0, s\_1)\in [ind]$ such that $(\Hm, s\_1) \models \varphi\_1$, \dots, $(\Hm, s\_1) \models \varphi\_n$, then there is $(s\_0, s\_1)\in R$ s.t. $(\Hm, s\_1) \models \bigwedge\_{i=1}^n \varphi\_i$, i.e. $(\Hm, s\_0) \models P\supset \EXIST \NEXT \bigwedge\_{i=1}^n \varphi\_i$.

% For each $(\Hm, s\_0) \in \Mod(P\supset \EXIST \NEXT \bigwedge\_{i=1}^n \varphi\_i)$, we suppose there is $(s\_0, s\_1)\in R$ s.t. $(\Hm, s\_1) \models \bigwedge\_{i=1}^n \varphi\_i$. It is easy to construct an initial \Ind-model $(\Hm', s\_0)$ such that $(\Hm', s\_0)$ is identical to $(\Hm, s\_0)$ except the $(s\_0, s\_1) \in [ind]$, i.e. $(\Hm, s\_0) \lrto\_{\tuple{\emptyset, \{ind\}}} (\Hm', s\_0)$.

% (ii) (If part) For any model $(\Hm,s\_0)$ of the left side of the equation if there is $(\Hm,s\_0) \models \bigwedge\_{i=1}^m P\_{j\_i}$ with $j\_i \in \{1, \dots, n\}$ and $1\leq m \leq n$, then there is a next state $s\_1$ of $s\_0$ with $(s\_0, s\_1) \in [ind]$ such that $(\Hm, s\_1) \models \bigwedge\_{i=1}^m \varphi\_{j\_i}$. By the definition of $[ind]$, we have $(s\_0, s\_1) \in R$ and then $(\Hm, s\_0) \models \bigwedge\_{i=1}^m P\_{j\_i} \supset \EXIST \NEXT (\bigwedge\_{i=1}^m P\_{j\_i} \varphi\_{j\_i})$. The other side can be similarly proved as (i).

% (iii) (Only if part) For any model $(\Hm,s\_0)$ of the right side of the equation if there is $(\Hm,s\_0) \models P$ then there exists a path $\pi\_{s\_0}$ such that $\varphi\_i \in \pi\_{s\_0}$ ($1\leq i \leq n$). This means we can construct an initial \Ind-model $(\Hm', s\_0)$ such that $(\Hm', s\_0)$ is identical to $(\Hm, s\_0)$ except for each $(s\_j, s\_{j+1})$ of $\pi\_{s\_0}$ there is $(s\_j, s\_{j+1}) \in [ind]$ $(0\leq j)$. It is easy to check $(\Hm', s\_0) \models \bigwedge\_{i=1}^n (P\supset \EXIST\_{\tuple{ind}} \FUTURE \varphi\_i)$ and $(\Hm, s\_0) \lrto\_{\tuple{\emptyset, \{ind\}}} (\Hm', s\_0)$. The other side can be shown similarly as in (ii).

% Other results can be proved similarly.

% \end{proof}

Observe that after using the equivalence in Proposition~\ref{pro:Ind:EF} to transform the $\EXIST\_{\tuple{ind}} \FUTURE$ into the form on the right-hand side, we can combine all the $\EXIST\_{\tuple{ind}} \NEXT$ clauses by using (i) and (ii) in Proposition~\ref{pro:In2NI},

and then remove all the $\tuple{ind}$.

Now assume that $\emph{RemA}$ is the set of clauses obtained by executing Removing\\_atoms. The idea is that $\emph{Removing\\_index}$ transforms (the formulas in) $\emph{RemA}$ to a set of formulas without the indexes by using the equations in Proposition~\ref{pro:In2NI}.

This means that we can transform the set clauses $\emph{RemA}$ into a set of formulae without indexes. This fact is formalised by the following result.

\begin{proposition}\label{lem:No:Ind}

%\textbf{(NI-BRemain)}

$\emph{RemA}\equiv\_{\tuple{\emptyset, I}} \emph{Removing\\_index}(\emph{RemA})$, where $I$ is the set of indexes in $\emph{RemA}$.

\end{proposition}

% \begin{proof}

% It is easy checking that from the definition of $\emph{Removing\\_index}$ and Proposition~\ref{pro:In2NI}.

% \end{proof}

Realise that we do not need such process in Example~\ref{examp:remA} since there was no index in the set of formulae.

Only thing that is left to deal is ``to remove" the $\start$ which is handled by the the following operator: Given a set $T$ of formulae:

\begin{align\*}

&T\_{\CTL} = \{C|C'\in T\ \mbox{and}\ C\ \mbox{is}\ D \ \mbox{if}\ C' \mbox{is the form}\\

& \ALL\GLOBAL(\start\supset D), \mbox{else}\ C\ \mbox{is}\ C' \ \mbox{itself}\}.

\end{align\*}

In such a case, one can see that $T \equiv T\_{\CTL}$ by $\varphi \equiv \ALL \GLOBAL (\start \supset \varphi)$~\cite{bolotov2000clausal}.

% In this way, we know that for any $\psi$ and $V$ if $\IR(\psi, V \cup V')$ then $\varphi \models \psi$ iff $\emph{ERes}(\varphi,V) \models \psi$, in which $\emph{ERes}(\varphi,V)$ is the result of Algorithm~\ref{alg:compute:forgetting:by:Resolution}.

% The last step of our algorithm is to eliminate all the atoms in $V'$ which has been introduced in the \emph{Transform} process.

% Let $\Gamma=\emph{Instantiate}(Res, V')$ and $\Gamma\_1=\emph{Removing\\_atoms} (\emph{Connect}(\Gamma))$, then $\emph{Replacing\\_atoms}(\emph{Removing\\_index}(\Gamma\_1))$ is obtained from $\emph{Removing\\_index}(\Gamma\_1)$ by doing the following three steps for each $p\in (V'\setminus \Gamma)$:

% \begin{itemize}

% \item replacing each $p\supset \varphi\_1\vee \dots \vee p \supset \varphi\_n$ with $p \supset \bigvee\_{i=1}^n \varphi\_i$;

% \item replacing $p\supset \varphi\_{1}\wedge \dots \wedge p \supset \varphi\_{m}$ with $\varphi\_j$ are instantiate formulae of $\Gamma$ $(j \in \{1,\dots, m\})$ with $p \supset \psi$, where $\psi=\bigwedge\_{j=1}^{m} \varphi\_{j}$ and $p$ do not appear in $\varphi\_j$, .

% \item For any formula $C\in \Gamma\_1$, replacing every $p$ in $C$ with $\psi$.

% \end{itemize}

% %Where $\NI(S)$ means do $\NI(e)$ for each $e\in S$ with $S$ is a set of sets of $\CTLsnf$ clause.

% Recall that any atom in $V'$ introduced in the Transform process is a name of the sub-formula of $\varphi$~\cite{bolotov2000clausal}. Apparently, this process is just a process of replacing each atom with an equivalent formula. Then we have:

% \begin{proposition}\label{pro:replaceA}

% Let $\Gamma\_1=\emph{Instantiate}(Res, V')$, $\Gamma\_2 =\emph{Removing\\_atoms}$ $(\emph{Connect}(\Gamma\_1), \Gamma\_1)$ and $\Gamma\_3 = \emph{Replacing\\_atoms}(\emph{Removing\\_index}(\Gamma\_2))$, then $\Gamma\_2 \equiv\_{\tuple{V'\setminus \Gamma\_1, I}} \Gamma\_3$ and $\varphi \equiv\_{\tuple{V\cup V', \emptyset}}$ $(\Gamma\_3)\_{CTL}$.

% \end{proposition}

% %\begin{proof}

% %For each $p$ talked above is a name of the formula $\psi$. %, \ie $p \lrto \psi$.

% %Then $\Gamma\_2 \equiv\_{\tuple{(V'\setminus \Gamma\_1), {\O}}} \Gamma\_3$, and then $\Gamma\_2 \equiv\_{\tuple{V\cup V', I}} \Gamma\_3$ by (V) of Proposition~\ref{div}.

% %

% %Therefore, $\varphi \equiv\_{\tuple{V\cup V',{\O}}} (\Gamma\_3)\_{CTL}$ by Proposition~\ref{pro:elm} and the definitions of $\emph{Removing\\_index}$ and $T\_{\CTL}$.

% %\end{proof}

% \begin{example}\label{exa:replace:sub}

% By using the \emph{Replacing\\_atoms} process on result of Example~\ref{examp:remA} directly since there is no index in those clauses, we obtain that $x$ is replaced by $f\vee m$. Then $y$ is replaced by $q \wedge \ALL\NEXT(q \vee f\vee m)$ and $z$ is replaced by $r\wedge (f\vee m \vee q) \wedge (f\vee m \vee (q\wedge \ALL\NEXT(f\vee m\vee q))) \wedge \ALL\FUTURE(f \vee m)$.

% \end{example}

% \subsection{An Example for the Connect Process}

% In order to show the necessity of the Connect process, we give the following example.

% \begin{example}

% Let $\psi=\ALL\FUTURE(p \wedge q) \wedge \EXIST \NEXT \neg p$ and $V=\{p\}$.

% By the processes Transform and Resolution, we can obtain $V'=\{f,z,w\}$ and the following set $Res$ of $\CTLsnf$ clauses.

% \begin{align\*}

% & \start \supset z && z \supset \ALL\FUTURE f && z \supset \EXIST\_{\tuple{ind}} \NEXT \neg p \\

% & \top \supset \neg f \vee p && \top \supset \neg f \vee q && z \supset \EXIST\_{\tuple{ind}} \NEXT \neg f

% \end{align\*}

% According to our Algorithm~\ref{alg:compute:forgetting:by:Resolution}, we have $\emph{Instantiate}(Res,V')=\{p,w\}$ since $f$ can be instantiated by $q$ and $z$ can be instantiated by $\ALL\FUTURE f$.

% On the one hand, in the \emph{Connect} process, by using \textbf{(EF1)} rule on the $Res$ we have $\alpha=z \supset (\neg q \supset (\EXIST\_{\tuple{ind}}\NEXT (q \supset \ALL\NEXT \ALL\FUTURE q )))$ and replace $z \supset \EXIST\_{\tuple{ind}} \NEXT \neg f \in Res$ with $z \supset \EXIST\_{\tuple{ind}} \NEXT \neg f \vee \alpha$ since $l$, $C\_2$, $C\_3$ and $C\_4$, which are $f$, $\Empty$, $q$ and $\Empty$ respectively ($\Empty$ express that there is not such clause), are instantiate formulae.

% Apparently, $z \supset \EXIST\_{\tuple{ind}} \NEXT \neg f \vee \alpha \equiv z \supset q \vee \EXIST\_{\tuple{ind}} \NEXT(\neg f \vee \neg q \vee \ALL\NEXT \ALL\FUTURE q)$.

% After the \emph{Removing\\_atoms} process, we have the following set \emph{RemA} of formulae:

% \begin{align\*}

% & \start \supset z && z \supset \ALL\FUTURE f \\

% & \top \supset \neg f \vee q && z \supset q \vee \EXIST\_{\tuple{ind}} \NEXT(\neg f \vee \neg q \vee \ALL\NEXT \ALL\FUTURE q)

% \end{align\*}

% Removing the indexes appearing in the \emph{RemA}, we obtain the following set $\NI$:

% \begin{align\*}

% & \start \supset z && z \supset \ALL\FUTURE f \\

% & \top \supset \neg f \vee q && z \supset q \vee \EXIST \NEXT(\neg f \vee \neg q \vee \ALL\NEXT \ALL\FUTURE q)

% \end{align\*}

% Replacing the atoms in $V'$ that have been instantiated, i.e. $f$ is replaced with $q$ and $z$ is replaced with $\ALL\FUTURE q \wedge (q \vee \EXIST \NEXT(\neg q \vee \ALL\NEXT \ALL\FUTURE q))$, we have

% \[

% \emph{Rp}= \{\start \supset \ALL\FUTURE q \wedge (q \vee \EXIST \NEXT(\neg q \vee \ALL\NEXT \ALL\FUTURE q))\}.

% \]

% As all the formulas $\cal F$ in the $T\_{\varphi}$ are the form $\ALL\GLOBAL \cal F$, hence we have:

% \[

% \emph{Rp}\_{\CTL} = \{\ALL\FUTURE q \wedge (q \vee \EXIST \NEXT(\neg q \vee \ALL\NEXT \ALL\FUTURE q))\}

% \]

% i.e. $\emph{ERes}(\varphi, V) = \ALL\FUTURE q \wedge (q \vee \EXIST \NEXT(\neg q \vee \ALL\NEXT \ALL\FUTURE q))$.

% In this case, we can easily check that $\emph{ERes}(\varphi, V) \equiv\_{\tuple{V, \emptyset}} \varphi$.

% On the other hand, if we do not using the \emph{Connect} process, we can easily obtain the result of $\emph{ERes}$, i.e. $\emph{ERes}(\varphi, V) = \ALL\FUTURE q \wedge \EXIST \NEXT(\neg q)$.

% It is apparent that $\emph{ERes}(\varphi, V) \not\equiv\_{\tuple{V, \emptyset}} \varphi$. This can proved by model $(\Hm,s\_0)$ as in Figure~\ref{Fig:models} since $(\Hm, s\_0) \models \varphi$ and $(\Hm, s\_0) \not \models \emph{ERes}(\varphi, V)$.

% \begin{figure}[ht!]

% \centering

% %Requires \usepackage{graphicx}

% \includegraphics[width=5cm]{models.png}\\

% \caption{A model $(\Hm, s\_0)$ of $\varphi$}\label{Fig:models}

% \end{figure}

% \end{example}

% This example shows why we introduce the \textbf{EF}-implication rules. Intuitively, the result of replacing the atoms that have been instantiated in $V'$ with an instantiate formula is stronger than our method, because by the \emph{Removing\\_atoms} process, we have removed some clauses, such as $C= \top \supset \neg f \vee p$, that contain $f$. The original one is $f \supset p \wedge q$, but after removing $C$ we only obtain $f \supset q$. In this example, there is a clause $z \supset \EXIST \NEXT \neg f \in Res$, after replacing $f$ with $q$, we obtain $z \supset \EXIST \NEXT \neg q$. However, if we do not remove $C$ (i.e. $f \supset p \wedge q$), then we have $z \supset \EXIST \NEXT (\neg q \vee \neg p)$, this is weaker than $z \supset \EXIST \NEXT \neg q$.

% In fact, for any model $(\Hm, s\_0)$ of $\varphi$, it might be the case that $q \in L(s)$ for all next states $s$ of $s\_0$ and if there is $q \in L(s)$ for all next states s, then there must be a next state $s$ of $s\_0$ with $p \not \in L(s)$ s.t. for all next state $s'$ of $s$ there is $(\Hm, s')\models \ALL\FUTURE q$ (see Fig.~\ref{Fig:models}).

%This is what the meaning of the \emph{Connect} process.

\subsection{Termination and Complexity of the Algorithm}

% \begin{proposition}

% Given a \CTL\ formula and any set $V$ of atoms, the Algorithm~\ref{alg:compute:forgetting:by:Resolution} will terminate.

% \end{proposition}

% \begin{proof}

% We can know that the Transform and Resolution can terminate from~\cite{zhang2009refined}. Moreover, the Remove\\_atoms and the Remove\\_index (inclue $T\_{\CTL}$) can also terminate because the set of clauses obtained from the Resolution process is finite.

% \end{proof}

We know that for every \CTL\ formula, the transformation and resolution processes terminate~\cite{zhang2014resolution}. Moreover, the Remove\\_atoms and the Remove\\_index (include $T\_{\CTL}$) processes also terminate because the set of clauses obtained from the resolution process is finite. Therefore, for any given \CTL\ formula and set $V$ of atoms, we can conclude that the Algorithm~\ref{alg:compute:forgetting:by:Resolution} will eventually terminate. However, the computational complexity of the algorithm is in question. We report the (time and space) complexity of the algorithm in the following result.

\begin{proposition}\label{pro:complexity}

Let $\varphi$ be a CTL formula and $V \subseteq \Ha$.

The time and space complexity of Algorithm~\ref{alg:compute:forgetting:by:Resolution} are $O((m+1)2^{4(n+n')}$ where $|\Var(\varphi)|=n$, $|V'|=n'$ ($V'$ is the set of atoms introduced in the transformation process) and $m$ is the number of indices introduced during transformation.

\end{proposition}

% \begin{proof}

% It follows from the lines 19-31 of the algorithm~\ref{alg:compute:forgetting:by:Resolution}, which is to compute all the possible resolution.

% The possible number of $\CTLsnf$ clauses under the give $V$, $V'$ and $Ind$ is $(m+1)2^{4(n+n')}+(m\*(n+n')+n+n'+1)2^{2(n+n')+1})$.

% \end{proof}

Indeed, observe that $m$ is at most as high as the number of temporal operators in $\varphi$.

This observation provides us with the insight that the computational complexity of our algorithm only depends on the number of atoms and temporal operators in $\varphi$.

% Although it is exponential, it is more efficient than that of the model-based algorithm in~\cite{renyansfirstpaper}, which not only depends on the number of atoms in $\Ha$ but also the number of states.

\subsection{Eliminating the atoms in $V'$}

Recall that we have eliminated the atoms in $V$, but the atoms in $V'$ that are introduced during the transformation process still remains. In order to eliminate them (as many as possible), we use the following generalisation~\footnote{We call it generalised, in the sense that, it also takes care of the temporal operators (in addition to classical Ackermann's Lemma).} of Ackermann's Lemma~\cite{szalas2002second} (before removing the propositional constant $\start$).

% \begin{lemma}[Ackermann-Lemma~\cite{szalas2002second}]\label{lem:ackl}

% Let $X$ be a relation variable and $\alpha(\overline{x}, \overline{z})$, $\beta(X)$ be classical first-order formulas, where the number of distinct variables in $\overline{x}$ is equal to the arity of $X$. Let $\alpha$ contain no occurrences of $X$.

% \begin{enumerate}[(i)]

% \item If $\beta(X)$ is positive w.r.t. $X$, then

% \[

% \exists X\{\forall\overline{x}[X(\overline{x}) \rto \alpha(\overline{x}, \overline{z})] \wedge \beta(X)\} \equiv \beta(X)\_{\alpha(\overline{x}, \overline{z})}^{X(\overline{x})}.

% \]

% \item If $\beta(X)$ is negative w.r.t. $X$, then

% \[

% \exists X\{\forall\overline{x}[\alpha(\overline{x}, \overline{z}) \rto X(\overline{x})] \wedge \beta(X)\} \equiv \beta(X)\_{\alpha(\overline{x}, \overline{z})}^{X(\overline{x})}.

% \]

% \end{enumerate}

% \end{lemma}

\begin{theorem}[Generalised Ackermann’s Lemma] \label{thm:Aclm}

Let $\Gamma$ be a set of formulae that contains the set $\Delta = \{\top \supset \neg x \vee C\_1$, \dots, $\top \supset \neg x \vee C\_n, x \supset B\_1, \dots, x \supset B\_m\}$ of clauses, where $x \in V'$ is an atom introduced in the transformation process, the $C\_i$ $(1 \leq i \leq n)$ are classical propositional clauses that do not contain $x$, and $B\_j$ ($1 \leq j \leq m$) are formulae of the disjunction (or conjunction) of formulae of form $Qt {\cal T} C$ with $Qt$ is empty (with ${\cal T}$ is empty) or $Qt \in \{\ALL, \EXIST\}$, ${\cal T}\in \{\NEXT, \FUTURE\}$ and $C$ is a CNF (or DNF) that also do not contain $x$. If $\Gamma'= \Gamma - \Delta$ is positive w.r.t. $x$ (i.e. each clause in $\Gamma'$ is positive w.r.t. $x$), then $\Gamma'[x/\varphi] \equiv\_{\tuple{\{x\}, \emptyset}} \Gamma$ with $\varphi = \bigwedge\_{i=1}^n C\_i \wedge \bigwedge\_{j=1}^m B\_j$, where $\Gamma'[x/\varphi]$ is obtained from $\Gamma'$ by replacing all $x$ with $\varphi$.

\end{theorem}

Informally, this result says that, such replacement (of every atom introduced in the transformation process in $V'$ with the appropriate CTL formula) is exactly the result of forgetting $x$ from $\Gamma$.

% \begin{proof}

% Without loss of generality, we suppose there are only A-step clauses in $\Gamma'$, other cases can be proved similarly.

% $(\Rto)$ For any model $(\Hm, s\_0)$ of $\Gamma$, it is obvious that $(\Hm, s\_0) \models \Gamma'$.

% $(\Lto)$ For any models $(\Hm, s\_0)$ of $\Gamma'$ with $\Hm = (S, R, L, s\_0)$, we can construct an \Ind-initial structure $\Hm'=(S', R', L', s\_0')$ with $S'=S$, $R'=R$, $s\_0'= s\_0$ and $L'$ is the same with $L$ except that for each $s'\in S'$ if $(\Hm', s') \models x \wedge \vaprhi$ then let $L'(s') = L(s) - \{x\}$, else let $L'(s') = L(s)$.

% It is easy to check that $(\Hm,s\_0) \lrto\_{\tuple{\{x\}, \emptyset}} (\Hm', s\_0')$ and $(\Hm',s\_0') \models \Gamma'$.

% \end{proof}

\begin{example}[cont'd from Example~3]

Recall that all the $\CTLsnf$ clauses are of the form $\ALL\GLOBAL(\psi\_1 \supset \psi\_2)$, in this case, the result of Example~\ref{examp:remA}, after using Theorem~\ref{thm:Aclm} on $x$ (i.e., $\Delta = \{\top \supset \neg x \vee f \vee m\}$, supposing that the set of clauses in Example~\ref{examp:remA} is $\Gamma$), can be expressed as follows:

\begin{align\*}

& \start\supset z && \start \supset f \vee m \vee y \\

& \top \supset \neg z \vee f \vee m \vee q && \top \supset \neg z \vee f \vee m \vee y \\

& \start \supset f\vee m \vee q && \top \supset \neg y \vee q\\

& z \supset \ALL \FUTURE (f \vee m) && y \supset \ALL \NEXT(f \vee m \vee y) \\

& y \supset \ALL\NEXT(f \vee m\vee q) &&

%& \start \supset f\vee m \vee q

\end{align\*}

Similarly, we can use Theorem~\ref{thm:Aclm} on $z$ where $\Delta=\{\top \supset \neg z \vee f \vee m \vee y, \top \supset \neg z \vee f \vee m\vee q, \top \supset \neg z \vee \ALL \FUTURE (f \vee m)\}$, and we get the following set $\NI$ of clauses:

\begin{align\*}

& \start\supset (f \vee m \vee y) \wedge (f \vee m\vee q) \wedge \ALL \FUTURE (f \vee m) && \\

& \start \supset f\vee m \vee q \qquad \top \supset \neg y \vee q\\

& y \supset \ALL \NEXT(f \vee m \vee y) \qquad \start \supset f \vee m \vee y \\

& y \supset \ALL\NEXT(f \vee m\vee q) \qquad

\end{align\*}

In this way, only $y\in V'$ is not eliminated.

\end{example}

Clearly, if all atoms in $V'$ are eliminated, then we can see that the result obtained from our algorithm is the result of forgetting.

We can also see that this process will help us to eliminate as many clauses as possible while not affecting the satisfiability of the original formula.

%Moreover, this process is helpful in transforming the result obtained after the Remove\\_atoms process.

% This means that if we only want to decide the satisfiability of forgetting $V$ from $\varphi$ by \CTL-RP, we do not need to do the Ackermann-Lemma and $T\_{\CTL}$ processes since the \CTL-RP always need to transform a \CTL\ into a set of $\CTLsnf$ clauses.

%\footnote{An additional remark is that if one uses the solver \CTL-RP (\url{https://sourceforge.net/projects/ctlrp/}), then there is no need to use the Ackermann-Lemma and the $T\_{\CTL}$ process since the \CTL-RP always need to transform a \CTL\ formula into a set of $\CTLsnf$ clauses.}

% In particular, if a \CTL\ formula is in NNF and do not contain the modal operators $Pt \GLOBAL$ and $Pt \UNTIL$ with $Pt \in \{\ALL, \EXIST\}$, then we can easily prove that we can eliminate all the introduced atoms in $V'$. The following example illustrates this point well.

% \begin{example}

% \end{example}

% Then it is clearly the $\bigwedge\_{\psi\in \NI\_{\CTL}} \psi$ is as follows:

% \begin{align\*}

% & r \wedge (f \vee m \vee y) \wedge (f \vee m\vee q) \wedge \ALL \FUTURE (f \vee m) \wedge \\

% & \ALL \GLOBAL( (\neg y \vee q) \wedge (y \supset \ALL\NEXT((f \vee m\vee q) \wedge (f \vee m \vee y)))).

% \end{align\*}

\section{Related Work}

Further related work, apart from \cite{renyansfirstpaper} which is explained in detail in the introduction, is as follows.

%\subsection{Forgetting}

%As a logical notion, \emph{forgetting} was first formally defined

%in propostional and first order-logics by Lin and Reiter~\cite{lin1994forget}, and then used to %compute the strongest necessary and weakest sufficient conditions in planning %\cite{DBLP:journals/ai/Lin01}.

%Over the last twenty years, not only have researchers developed forgetting notions and theories in %propositional and first-order logic, but also in other logic systems~\cite{eiter2019brief},

%such as in logic programs under answer set/stable model %semantics~\cite{DBLP:Zhang:AIJ2006,Eiter2008Semantic,Wong:PhD:Thesis,Yisong:KR:2012,Yisong:IJCAI:2%013}, forgetting in description logic to create restricted views of %ontologies~\cite{Wang:AMAI:2010,Lutz:IJCAI:2011,zhao2017role} and in modal %logic~\cite{Yan:AIJ:2009,Kaile:JAIR:2009,Yongmei:IJCAI:2011,fang2019forgetting}.

%Moreover, forgetting has also been used in conflict solving %\cite{Lang2010Reasoning,Zhang2005Solving} and

%strongest and weakest definitions \cite{Lang2008On}.

%knowledge compilation \cite{Zhang2009Knowledge,Bienvenu2010Knowledge},

%creating restricted views of ontologies~\cite{zhao2017role},

%{ZhaoSchmidt18a},

% strongest and weakest definitions \cite{Lang2008On}, SNC (WSC) \cite{DBLP:journals/ai/Lin01} and others.

%\noindent \textbf{Resolution-based satisfiability in \CTL:}

%Deciding satisfiability with resolution calculus in Propositional Linear Temporal Logic (PLTL) was introduced in~\cite{fisher1991resolution} and further discussed in~\cite{fisher1997normal,fisher2001clausal}.

% The main idea is to transform PLTL formulas into the a normal form, called Separated Normal Form (SNF) by introducing a new connective \start\ that holds only at the beginning of time.

Resolution-based satisfiability in \CTL\ was proposed by Bolotov in~\cite{bolotov2000clausal} and then further refined by Zhang in~\cite{zhang2014resolution}.

In those papers, the main idea is to transform \CTL\ formulas into a normal form $\CTLsnf$.

But since \CTL\ is a type of branching time temporal logic, they introduce ``indices" besides \start\ for that purpose.

Other resolution procedures which have been used to compute the forgetting or uniform interpretation in propositional logic are \cite{Yisong:2015:arx} and in modal logic~\cite{herzig2008uniform}. In those cases, the formula is required to be in CNF form~\cite{herzig2008uniform}).

% As aforementioned, the normal form used here for resolution is an extension of \CTL\ with \start\ and ``index".

%In this article, we propose $\tuple{V,I}$-bisimulation to deal with the ``index" problem.

%In order to eliminate those atoms introduced in the transformation, we proposed the four EF-implication rules.

As the case in~\cite{zhao2018automated}, our algorithm also introduces the new atoms and it is not guaranteed that $\emph{ERes}(\varphi,V)$ is exactly equivalent to the result of forgetting $V$ from $\varphi$, however the atoms in $V$ can all be forgotten and if the atoms in $V'$ are all eliminated too, then $\emph{ERes}(\varphi,V) \equiv \CTLforget(\varphi, V)$. In this sense we can improve the performance of deciding the satisfiability of a \CTL\ formula by using the solver CTL-RP since some clauses which contain atoms in $V$ would be removed (i.e., have fewer $\CTLsnf$ clauses).

%Moreover, for the resolution-based satisfiability solver \CTL-RP of \CTL\ formulae, if we can eliminate some atoms of a formula in advance or , then it will be faster to decide the satisfiability of a formula.

\section{Conclusion and Future Work}

In this paper, we proposed a resolution-based algorithm to compute the forgetting in \CTL.

Our method extend the resolution calculus in~\cite{zhang2014resolution} by adding processes that remove irrelevant atoms and transforming the result back to a \CTL\ formula.

For this purpose, we defined a new type of binary bisimulation relation, called $\tuple{V,I}$-bisimulation, in order to bridge the gap between \CTL\ and $\CTLsnf$. Besides, for eliminating the irrelevant atoms we proposed the Removing\\_atoms operator and the generalised Ackermann’s Lemma. The Removing\\_atoms operator can exactly eliminate all the atoms in $V$, and the generalised Ackermann’s Lemma help us eliminate the atoms in $V'$ as many as possible.

Finally, we proved that our algorithm terminates, and if all the atoms in $V'$ can be eliminated then our algorithm will return the result of forgetting $V$ from a \CTL\ formula.

%, and the time and space complexity of Algorithm~\ref{alg:compute:forgetting:by:Resolution} are $O((m+1)2^{4(n+n')}$.

Considering the computational complexity result, our resolution-based method is way more efficient than the model-based approach reported in~\cite{renyansfirstpaper}.

%In the future we will implement this algorithm (part of it has been implemented actually).

As for the future research, a Prolog implementation is currently under development, and we are planning to evaluate its practical aspects.\footnote{We do not share the anonymised link due to recent submission regulations (it will be made available in case of an acceptance).}

Moreover, as for the theory, we are planning to carry out a parameterised complexity analysis on our resolution calculus.

%Besides, applying the defined methods to similar formalism.

%\section\*{Acknowledgments}

\clearpage

\bibliographystyle{aaai21}

\bibliography{Ref}

\clearpage

\appendix

\section{Supplementary Material: Proof Appendix}

% \begin{lemma}\label{lem:B:relations}

% Let $\Hb\_0, \Hb\_1,\ldots$ be the ones in the definition of section 3.1.

% Then, for each $i\ge 0$,

% \begin{enumerate}[(i)]

% \item $\Hb\_{i+1}\subseteq \Hb\_i$;

% \item there is a (smallest) $k\ge 0$ such that $\Hb\_{k+1}=\Hb\_k$;

% \item $\Hb\_i$ is reflexive, symmetric and transitive.

% \end{enumerate}

% \end{lemma}

% \begin{proof}

% See~\cite{renyansfirstpaper}.

% % (i)

% % Base: it is clear for $i=0$ by the above definition.

% % Step: suppose it holds for $i=n$, i.e., $\Hb\_{n+1}\subseteq\Hb\_n$. \\

% % $(s,s')\in\Hb\_{n+2}$\\

% % $\Rto$ (a) $(s,s')\in \Hb\_0$,

% % (b) for every $(s,s\_1)\in R$, there is $(s',s\_1')\in R'$

% % such that $(s\_1,s\_1')\in \Hb\_{n+1}$, and

% % (c) for every $(s',s\_1')\in R'$, there is $(s,s\_1)\in R$

% % such that $(s\_1,s\_1')\in \Hb\_{n+1}$\\

% % $\Rto$ (a) $(s,s')\in \Hb\_0$,

% % (b) for every $(s,s\_1)\in R$, there is $(s',s\_1')\in R'$

% % such that $(s\_1,s\_1')\in \Hb\_{n}$ by inductive assumption, and

% % (c) for every $(s',s\_1')\in R'$, there is $(s,s\_1)\in R$

% % such that $(s\_1,s\_1')\in \Hb\_{n}$ by inductive assumption\\

% % $\Rto$ $(s,s')\in \Hb\_{n+1}$.

% % (ii) and (iii) are evident from (i) and the definition of $\Hb\_i$.

% \end{proof}

% \noindent\textbf{Lemma}~\ref{lem:equive} The relation $\lrto\_V$ is an equivalence relation.

% \begin{proof}

% See~\cite{renyansfirstpaper}.

% % It is clear from Lemma~\ref{lem:B:relations} (ii) such that there is a $k \geq $ 0 where $\Hb\_k = \Hb\_{k+1}$ which is $\lrto\_V$, and it is reflexive, symmetric and transitive by (iii).

% \end{proof}

% \noindent\textbf{Proposition}~\ref{div}

% Let $i\in \{1,2\}$, $V\_1,V\_2\subseteq\cal A$

% and ${\cal K}\_i=({\cal M}\_i,s\_i)~(i=1,2,3)$ be \MPK-structures (Ind-structures)

% such that

% ${\cal K}\_1\lrto\_{V\_1}{\cal K}\_2$ and ${\cal K}\_2\lrto\_{V\_2}{\cal K}\_3$.

% Then:

% \begin{enumerate}[(i)]

% \item ${\cal K}\_1\lrto\_{V\_1\cup V\_2}{\cal K}\_3$;

% \item If $V\_1 \subseteq V\_2$ then ${\cal K}\_1 \lrto\_{V\_2} {\cal K}\_2$.

% \end{enumerate}

% \begin{proof}

% See~\cite{renyansfirstpaper}.

% % In order to distinguish the relations $\Hb\_0, \Hb\_1, \dots$ for different set $V \subseteq \Ha$, by $\Hb\_i^V$ we mean the relation $\Hb\_1, \Hb\_2, \dots$ for $V \subseteq \Ha$.

% % Denote as $\Hb\_0, \Hb\_1, \dots$ when the underlying set $V$ is clear from the context. Moreover, for the ease of notation, we will refer to $\lrto\_V$ by $\Hb$ (i.e., without subindex).

% % The following property show our result directly.

% % Let $V\subseteq\cal A$

% % %${\cal M}\_i=(S\_i,R\_i,L\_i,s\_0^i)~(i=1,2)$ be Kripke structures

% % and ${\cal K}\_i=({\cal M}\_i,s\_i)~(i=1,2)$ be \MPK-structures.

% % Then $({\cal K}\_1,{\cal K}\_2)\in\cal B$ if and only if

% % \begin{enumerate}[(a)]

% % \item $L\_1(s\_1)- V = L\_2(s\_2)- V$,

% % \item for every $(s\_1,s\_1')\in R\_1$, there is $(s\_2,s\_2')\in R\_2$

% % such that $({\cal K}\_1',{\cal K}\_2')\in \Hb$, and

% % \item for every $(s\_2,s\_2')\in R\_2$, there is $(s\_1,s\_1')\in R\_1$

% % such that $({\cal K}\_1',{\cal K}\_2')\in \Hb$,

% % \end{enumerate}

% % where ${\cal K}\_i'=({\cal M}\_i,s\_i')$ with $i\in\{1,2\}$.

% % We prove it from the following two aspects:

% % $(\Rto)$

% % (a) It is apparent that $L\_1(s\_1)- V = L\_2(s\_2)- V$;

% % (b) %We will show that for each $(s\_1, s\_1') \in R\_1$, there is a $(s\_2, s\_2')\in R\_2$ such that $({\cal K}\_1', {\cal K}\_2') \in \Hb$.

% % $({\cal K}\_1, {\cal K}\_2) \in \Hb$ iff $({\cal K}\_1, {\cal K}\_2) \in \Hb\_i$ for all $i \geq 0$, then for each $(s\_1, s\_1') \in R\_1$, there is a $(s\_2, s\_2')\in R\_2$ such that $({\cal K}\_1', {\cal K}\_2') \in \Hb\_{i-1}$ for all $i > 0$ and then $L\_1(s\_1')- V = L\_2(s\_2')- V$. Therefore, $({\cal K}\_1', {\cal K}\_2') \in \Hb$.

% % (c) %We will show that for each $(s\_2, s\_2') \in R\_1$, there is a $(s\_1, s\_1')\in R\_2$ such that $({\cal K}\_1', {\cal K}\_2') \in \Hb$.

% % This is similar with (b).

% % $(\Lto)$ Apparently, $L\_1(s\_1)- V = L\_2(s\_2)- V$ implies that $(s\_1, s\_2) \in \Hb\_0$;

% % (b) implies that for every $(s\_1,s\_1')\in R\_1$, there is $(s\_2,s\_2')\in R\_2$

% % such that $({\cal K}\_1',{\cal K}\_2')\in \Hb\_i$ for all $i \geq 0$;

% % (c) implies that for every $(s\_2,s\_2')\in R\_2$, there is $(s\_1,s\_1')\in R\_1$

% % such that $({\cal K}\_1',{\cal K}\_2')\in \Hb\_i$ for all $i \geq 0$\\

% % $\Rto$ $({\cal K}\_1, {\cal K}\_2) \in \Hb\_i$ for all $i \geq 0$\\

% % $\Rto$ $({\cal K}\_1,{\cal K}\_2)\in\cal B$.

% % (i) Let ${\cal M}\_i=(S\_i,R\_i,L\_i,s\_i)~(i=1,2,3)$, $s\_1 \lrto\_{V\_1} s\_2$ via a binary relation $\Hb$, and $s\_2 \lrto\_{V\_2} s\_3$ via a binary relation $\Hb''$. Let $\Hb' = \{(w\_1, w\_3)| (w\_1, w\_2)\in \Hb$ and $(w\_2, w\_3)\in \Hb\_2\}$. It's apparent that $(s\_1, s\_3) \in \Hb'$. We prove $\Hb'$ is a $V\_1 \cup V\_2$-bisimulation containing $(s\_1, s\_3)$ from the (a), (b) and (c) of the previous steps of $X$-bisimulation (where $X$ is a set of atoms). For all $(w\_1, w\_3) \in \Hb'$:

% % \begin{enumerate}[(a)]

% % \item there is $w\_2 \in S\_2$ such that $(w\_1,w\_2)\in \Hb$ and $(w\_2, w\_3)\in \Hb''$, and $\forall q \notin V\_1$, $q \in L\_1(w\_1)$ iff $q \in L\_2(w\_2)$ by $w\_1 \lrto\_{V\_1} w\_2$ and $\forall q' \notin V\_2$, $q'\in L\_2(w\_2)$ iff $q'\in L\_3(w\_3)$ by $w\_2 \lrto\_{V\_2} w\_3$. Then we have $\forall r\notin V\_1 \cup V\_2$, $r \in L\_1(w\_1)$ iff $r \in L\_3(w\_3)$.

% % \item if $(w\_1, u\_1) \in \Hr\_1$, then $\exists u\_2\in S\_2$ such that $(w\_2, u\_2) \in \Hr\_2$ and $(u\_1,u\_2)\in \Hb$ (due to $(w\_1,w\_2)\in \Hb$ and $(w\_2, w\_3) \in \Hb''$ by the definition of $\Hb'$); and then $\exists u\_3 \in S\_3$ such that $(w\_3, u\_3) \in \Hr\_3$ and $(u\_2, u\_3) \in \Hb''$, hence $(u\_1, u\_3) \in \Hb'$ by the definition of $\Hb'$.

% % \item if $(w\_3, u\_3) \in \Hr\_3$, then $\exists u\_2\in S\_2$ such that $(w\_2, u\_2) \in \Hr\_2$ and $(u\_2, u\_3) \in \Hb\_2$; and then $\exists u\_1 \in S\_1$ such that $(w\_1, u\_1) \in \Hr\_1$ and $(u\_1, u\_2) \in \Hb$, hence $(u\_1, u\_3) \in \Hb'$ by the definition of $\Hb'$.

% % \end{enumerate}

% % (ii) Let ${\cal K}\_{i, j}=(\Hm\_i, s\_{i,j})$ and $(s\_{i, k}, s\_{i, k+1}) \in R\_i$ mean that $s\_{i, k+1}$ is the $(k+2)$-th node in the path

% % $(s\_i, s\_{i, 1}, s\_{i,2}, \dots , s\_{i, k+1}, \dots)$ ($i=1,2$).

% % We will show that $({\cal K}\_1, {\cal K}\_2) \in \Hb\_n^{V\_2}$ for all $n \ge 0$ inductively.

% % Base: $L\_1(s\_1) - V\_1 = L\_2(s\_2) - V\_1$\\

% % $\Rto$ $\forall q \in {\cal A} - V\_1$ there is $q \in L\_1(s\_1)$ iff $q \in L\_2(s\_2)$\\

% % $\Rto$ $\forall q \in {\cal A} - V\_2$ there is $q \in L\_1(s\_1)$ iff $q \in L\_2(s\_2)$ due to $V\_1 \subseteq V\_2$\\

% % $\Rto$ $L\_1(s\_1) - V\_2 = L\_2(s\_2) - V\_2$, i.e.,\ $({\cal K}\_1, {\cal K}\_2) \in \Hb\_0^{V\_2}$.

% % Step: Supposing that $({\cal K}\_1, {\cal K}\_2) \in \Hb\_i^{V\_2}$ for all $0 \leq i \leq k$ ($k > 0)$, we will show $({\cal K}\_1, {\cal K}\_2) \in \Hb\_{k+1}^{V\_2}$.

% % \begin{enumerate} [(a)]

% % \item It is apparent that $L\_1(s\_1) - V\_2 = L\_2(s\_2) - V\_2$ by base.

% % \item $\forall (s\_1, s\_{1,1}) \in R\_1$, we will show that there is a $(s\_2, s\_{2, 1}) \in R\_2$ s.t.\ $({\cal K}\_{1,1}, {\cal K}\_{2,1})\in \Hb\_k^{V\_2}$. $({\cal K}\_{1,1}, {\cal K}\_{2,1})\in \Hb\_{k-1}^{V\_2}$ by inductive assumption, we need only to prove the following points:\\

% % (a) $\forall (s\_{1, k}, s\_{1, k+1}) \in R\_1$ there is a $(s\_{2, k}, s\_{2, k+1})\in R\_2$ s.t.\ $({\cal K}\_{1,k+1}, {\cal K}\_{2,k+1})\in \Hb\_0^{V\_2}$ due to $({\cal K}\_{1,1}, {\cal K}\_{2,1})\in \Hb\_{k}^{V\_1}$. It is easy to see that $L\_1(s\_{1, k+1}) - V\_1 = L\_1(s\_{2, k+1}) - V\_1$, then there is $L\_1(s\_{1, k+1})- V\_2 = L\_1(s\_{2, k+1}) - V\_2$. Therefore, $({\cal K}\_{1,k+1}, {\cal K}\_{2,k+1})\in \Hb\_0^{V\_2}$.\\

% % (b) $\forall (s\_{2, k}, s\_{2, k+1}) \in R\_1$ there is a $(s\_{1, k}, s\_{1, k+1}) \in R\_1$ s.t.\ $({\cal K}\_{1,k+1}, {\cal K}\_{2,k+1})\in \Hb\_0^{V\_2}$ due to $({\cal K}\_{1,1}, {\cal K}\_{2,1})\in \Hb\_{k}^{V\_1}$. This can be proved as (a).

% % \item $\forall (s\_2, s\_{2,1}) \in R\_1$, we will show that there is a $(s\_1, s\_{1, 1}) \in R\_2$ s.t.\ $({\cal K}\_{1,1}, {\cal K}\_{2,1})\in \Hb\_k^{V\_2}$. This can be proved as (ii).

% % \end{enumerate}

% \end{proof}

% \noindent\textbf{Theorem}\ref{thm:V-bisimulation:EQ}

% Let $V\subseteq\cal A$, ${\cal K}\_i~(i=1,2)$ be two \MPK-structures such that

% ${\cal K}\_1\lrto\_V{\cal K}\_2$ and $\phi$ a formula with $\IR(\phi,V)$. Then

% ${\cal K}\_1\models\phi$ if and only if ${\cal K}\_2\models\phi$.

% \begin{proof}

% See~\cite{renyansfirstpaper}.

% % This theorem can be proved by inducting on the formula $\phi$ and supposing $\Var(\phi) \cap V = \Empty$.

% % Let ${\cal K}\_1 = (\Hm, s)$ and ${\cal K}\_2 = (\Hm', s')$.

% % %Here we only prove the only-if direction. The other direction can be similarly proved.

% % \textbf{Case} $\phi = p$ where $p \in \Ha - V$:\\

% % $(\Hm, s) \models \phi$ iff $p\in L(s)$ \hfill (by the definition of satisfiability) \\

% % $\LRto$ $p \in L'(s')$ \hfill ($s \lrto\_V s'$)\\

% % $\LRto$ $(\Hm', s') \models \phi$

% % \textbf{Case} $\phi = \neg \psi$:\\

% % $(\Hm, s) \models \phi$ iff $(\Hm, s) \nvDash \psi$ \\

% % $\LRto$ $(\Hm', s') \nvDash \psi$ \hfill (induction hypothesis)\\

% % $\LRto$ $(\Hm', s') \models \phi$

% % \textbf{Case} $\phi = \psi\_1 \vee \psi\_2$:\\

% % $(\Hm, s) \models \phi$\\

% % $\LRto$ $(\Hm, s) \models \psi\_1$ or $(\Hm, s) \models \psi\_2$\\

% % $\LRto$ $(\Hm', s') \models \psi\_1$ or $(\Hm', s') \models \psi\_2$ \hfill (induction hypothesis)\\

% % $\LRto$ $(\Hm', s') \models \phi$

% % \textbf{Case} $\phi = \EXIST \NEXT \psi$:\\

% % %By Lemma~\ref{V\_path}, we assume there are two paths $\pi = s, s\_1, ...$ and $\pi' = s', s\_1', ...$ such that $\pi \lrto\_V \pi'$.\\

% % $\Hm, s \models \phi$ \\

% % $\LRto$ There is a path $\pi = (s, s\_1, ...)$ such that $\Hm, s\_1 \models \psi$\\

% % $\LRto$ There is a path $\pi' = (s', s\_1', ...)$ such that $\pi \lrto\_V \pi'$ \hfill ($s \lrto\_V s'$, Proposition~\ref{div})\\

% % $\LRto$ $s\_1 \lrto\_V s\_1'$ \hfill ($\pi \lrto\_V \pi'$)\\

% % $\LRto$ $(\Hm', s\_1') \models \psi$ \hfill (induction hypothesis)\\

% % $\LRto$ $(\Hm', s') \models \phi$

% % \textbf{Case} $\phi = \EXIST \GLOBAL \psi$:\\

% % $\Hm, s \models \phi$ \\

% % $\LRto$ There is a path $\pi =(s=s\_0, s\_1, ...)$ such that for each $i \geq 0$ there is $(\Hm, s\_i) \models \psi$\\

% % $\LRto$ There is a path $\pi' = (s'=s\_0', s\_1', ...)$ such that $\pi \lrto\_V \pi'$ \hfill ($s \lrto\_V s'$, Proposition~\ref{div})\\

% % $\LRto$ $s\_i \lrto\_V s\_i'$ for each $i \geq 0$ \hfill ($\pi \lrto\_V \pi'$)\\

% % $\LRto$ $(\Hm', s\_i') \models \psi$ for each $i \geq 0$ \hfill (induction hypothesis)\\

% % $\LRto$ $(\Hm', s') \models \phi$

% % \textbf{Case} $\phi = \EXIST [\psi\_1 \UNTIL \psi\_2]$:\\

% % %\textbf{Case} $\varphi = \MPE \FUTURE \psi$:

% % $\Hm, s \models \phi$ \\

% % $\LRto$ There is a path $\pi= (s=s\_0, s\_1, ...)$ such that there is $i \geq 0$ such that $(\Hm, s\_i) \models \psi\_2$, and for all $0 \leq j < i$, $(\Hm, s\_j) \models \psi\_1$\\

% % $\LRto$ There is a path $\pi' = (s=s\_0', s\_1', ...)$ such that $\pi \lrto\_V \pi'$ \hfill ($s \lrto\_V s'$, Proposition~\ref{div})\\

% % $\LRto$ $(\Hm', s\_i') \models \psi\_2$, and for all $0 \leq j < i$ $(\Hm', s\_j') \models \psi\_1$ \hfill (induction hypothesis)\\

% % $\LRto$ $(\Hm', s') \models \phi$

% \end{proof}

\noindent\textbf{Proposition}~\ref{pro:VI:div}

Let $i\in \{1,2\}$, $V\_1,V\_2\subseteq\cal A$, $I\_1, I\_2 \subseteq \Ind$

and ${\cal K}\_i=({\cal M}\_i,s\_0^i)~(i=1,2,3)$ be initial Ind-structures

such that

${\cal K}\_1\lrto\_{\tuple{V\_1, I\_1}}{\cal K}\_2$ and ${\cal K}\_2\lrto\_{\tuple{V\_2,I\_2}}{\cal K}\_3$.

Then:

\begin{enumerate}[(i)]

% \item $s\_1'\lrto\_{V\_i}s\_2'~(i=1,2)$ implies $s\_1'\lrto\_{V\_1\cup V\_2}s\_2'$;

% \item $\pi\_1'\lrto\_{V\_i}\pi\_2'~(i=1,2)$ implies $\pi\_1'\lrto\_{V\_1\cup V\_2}\pi\_2'$;

% \item for each path $\pi\_{s\_1}$ of $\Hm\_1$ there is a path $\pi\_{s\_2}$ of $\Hm\_2$ such that $\pi\_{s\_1} \lrto\_{V\_1} \pi\_{s\_2}$, and vice versa;

\item ${\cal K}\_1\lrto\_{\tuple{V\_1\cup V\_2, I\_1 \cup I\_2}}{\cal K}\_3$;

\item If $V\_1 \subseteq V\_2$ and $I\_1 \subseteq I\_2$ then ${\cal K}\_1 \lrto\_{\tuple{V\_2, I\_2}} {\cal K}\_2$.

\end{enumerate}

\begin{proof}

%This can be proved similarly with Proposition~\ref{div}.

(i) By Proposition 1 in~\cite{renyansfirstpaper} we have ${\cal K}\_1\lrto\_{V\_1\cup V\_2}{\cal K}\_3$. For (i) of Definition~\ref{def:VInd:bisimulation} we can prove it as follows:

for all $(s,s\_1) \in [j]\_1$ there is a $(s', s\_1') \in [j]\_2$ such that $s\lrto\_{V\_1} s'$ and $s\_1 \lrto\_{V\_1} s\_1'$ and there is a $(s'', s\_1'') \in [j]\_3$ such that $s'\lrto\_{V\_2} s''$ and $s\_1' \lrto\_{V\_2} s\_1''$, then we have for all $(s,s\_1) \in [j]\_1$ there is a $(s'', s\_1'') \in [j]\_3$ such that $s \lrto\_{V\_1\cup V\_2} s''$ and $s\_1 \lrto\_{V\_1\cup V\_2} s\_1''$. The (ii) of Definition~\ref{def:VInd:bisimulation} can be proved similarly.

(ii) This can be proved from (ii) of Proposition 1 in~\cite{renyansfirstpaper}.

\end{proof}

\noindent\textbf{Proposition}~\ref{pro:TranE}

Let $\varphi$ be a \CTL\ formula, then $\varphi \equiv\_{\tuple{V', I}} T\_{\varphi}$.

\begin{proof} (sketch)

This can be proved from $T\_i$ to $T\_{i+1}$ $(0\leq i < n)$ by using one transformation rule on $T\_i$.

We will prove this proposition from the following several aspects:

(1) $\varphi \equiv\_{\tuple{\{p\}, {\O}}} T\_0$.

$(\Rto)$ For all $(\Hm\_1,s\_1) \in \Mod(\varphi)$, \ie $(\Hm\_1,s\_1) \models \varphi$. We can construct an \Ind-Kripke structure $\Hm\_2$ is identical to $\Hm\_1$ except $L\_2(s\_2) = L\_1(s\_1) \cup \{p\}$. It is apparent that $(\Hm\_2,s\_2) \models T\_0$ and $(\Hm\_1, s\_1) \lrto\_{\tuple{\{p\}, {\O}}} (\Hm\_2, s\_2)$.

$(\Lto)$ For all $(\Hm\_1,s\_1) \in \Mod(T\_0)$, it is apparent that $(\Hm\_1,s\_1) \models \varphi$ by the sematic of $\start$.

By $\psi \rto\_t R\_i$ we mean using transformation rules $t$ on formula $\psi$ (the formulae $\psi$ as the

premises of rule $t$) and obtaining the set $R\_i$ of its results. Let $X$ be a set of formulas,

we will show $T\_i \equiv\_{\tuple{V',I}} T\_{i+1}$ by using the transformation rule $t$. Where $T\_i= X \cup \{\psi\}$, $T\_{i+1}=X \cup R\_i$, $V'$ is the set of atoms introduced by $t$ and $I$ is the set of indexes introduced by $t$. (We will prove this result in $t\in \{$Trans(1), Trans(4), Trans(6)$\}$, other cases can be proved similarly.)

(2) For $t$=Trans(1):\\

$(\Rto)$ For all $(\Hm\_1,s\_1) \in \Mod(T\_i)$ \ie $(\Hm\_1, s\_1) \models X \wedge \ALL\GLOBAL(q \supset \EXIST \NEXT \varphi)$\\

$\Rto$ $(\Hm\_1,s\_1)\models X$ and for every $\pi$ starting from $s\_1$ and every state $s\_1^j \in \pi$, $(\Hm,s\_1^j) \models \neg q$ or there exists a path $\pi'$ starting from $s\_1^j$ such that $(s\_1^j,s\_1^{j+1})\in R\_1$ and $(\Hm,s\_1^{j+1})\models \varphi$.\\

We can construct an \Ind-Kripke structure $\Hm\_2$ is identical to $\Hm\_1$ except $[ind]\_2= \bigcup\_{s\in S} R\_s \cup R\_y$, where $ind$ is the index introduced by using Trans(1) on clause $\ALL\GLOBAL(q \supset \EXIST \NEXT \varphi)$, $R\_{s\_1^{j}}=\{(s\_1^{j},s\_1^{j+1}), (s\_1^{j+1}, s\_1^{j+2}),\dots\}$ and $R\_y=\{(s\_x,s\_y)| \text{ for all } s\_x \in S$ if for all $(s\_1',s\_2')\in \bigcup\_{s\in S} R\_s, s\_1'\neq s\_x$ then find exactly one state $s\_y\in S$ such that $(s\_x,s\_y)\in R\}$, which means for every $s\in S$ there exists exactly a state $s'\in S$ such that $(s,s')\in [ind]$ and $(s,s')\in R$. It is apparent that $(\Hm\_1, s\_1) \lrto\_{\tuple{{\O}, \{ind\}}} (\Hm\_2, s\_2)$ (let $s\_2=s\_1$).\\

$\Rto$ for every path starting from $s\_1$ and every state $s\_1^j$ in this path, $(\Hm\_2, s\_1^j) \models \neg q$ or $(\Hm\_2, s\_1^j)\models \EXIST \NEXT \varphi\_{\tuple{ind}}$ \hfill (by the semantic of $\EXIST \NEXT$)\\

$\Rto$ $(\Hm\_2, s\_1) \models \ALL \GLOBAL(q \supset \EXIST\_{\tuple{ind}} \NEXT \varphi )$\\

$\Rto$ $(\Hm\_2, s\_1) \models X \wedge \ALL \GLOBAL(q \supset \EXIST\_{\tuple{ind}} \NEXT \varphi )$

$(\Lto)$ For all $(\Hm\_1,s\_1) \in \Mod(T\_{i+1})$ \ie $(\Hm\_1,s\_1) \models X \wedge \ALL \GLOBAL(q \supset \EXIST\_{\tuple{ind}} \NEXT \varphi )$\\

$\Rto$ $(\Hm\_1,s\_1) \models X$ and $(\Hm\_1,s\_1) \models \ALL \GLOBAL(q \supset \EXIST\_{\tuple{ind}} \NEXT \varphi)$\\

$\Rto$ for every path starting from $s\_1$ and every state $s\_1^j$ in this path, $(\Hm\_1, s\_1^j) \models \neg q$ or there exits a state $s'$ such that $(s\_1^j, s')\in [ind]\_1$ and $(\Hm\_1, s') \models \varphi$ \hfill (by the semantic of $\EXIST\_{\tuple{ind}} \NEXT$)\\

$\Rto$ for every path starting from $s\_1$ and every state $s\_1^j$ in this path, $(\Hm\_1, s\_1^j) \models \neg q$ or $(\Hm\_1, s\_1^j) \models \EXIST \NEXT \varphi$ \hfill (by the semantic of $\EXIST \NEXT$)\\

$\Rto$ $(\Hm\_1,s\_1) \models \ALL\GLOBAL(q \supset \EXIST \NEXT \varphi)$\\

$\Rto$ $(\Hm\_1, s\_1) \models X \wedge \ALL\GLOBAL(q \supset \EXIST \NEXT \varphi)$\\

It is apparent that $(\Hm\_1, s\_1) \lrto\_{\tuple{{\O}, \{ind\}}} (\Hm\_1, s\_1)$.

(3) For $t$=Trans(4):\\

$(\Rto)$ For all $(\Hm\_1,s\_1) \in \Mod(T\_i)$, \ie $(\Hm\_1,s\_1) \models X \wedge \ALL\GLOBAL (q \supset \varphi\_1 \vee \varphi\_2)$ \\

$\Rto$ $(\Hm\_1,s\_1) \models X$ and $\forall s\_1'\in S, (\Hm\_1,s\_1') \models q \supset \varphi\_1 \vee \varphi\_2$\\

$\Rto$ $(\Hm\_1,s\_1') \models \neg q$ or $(\Hm\_1,s\_1') \models \varphi\_1 \vee \varphi\_2$\\

The we can construct an \Ind-Kripke structure $\Hm\_2$ as follows: $\Hm\_2$ is the same with $\Hm\_1$ except for each state $s\_1'$ if $(\Hm\_1,s\_1') \models \neg q$ then $L\_2(s\_1')= L\_1(s\_1')$, else if $(\Hm\_1,s\_1') \models \varphi\_1$ then $L\_2(s\_1')= L\_1(s\_1')$ else $L\_2(s\_1') = L\_1(s\_1') \cup \{p\}$. It is apparent that $(\Hm\_2,s\_1') \models (q\supset \varphi\_1 \vee p) \wedge (p \supset \varphi\_2)$ and $(\Hm\_1, s\_1) \lrto\_{\tuple{\{p\}, {\O}}} (\Hm\_2, s\_2)$, then $(\Hm\_2,s\_1) \models T\_{i+1}$.

$(\Lto)$ For all $(\Hm\_1, s\_1) \in \Mod(T\_{i+1})$, \ie $(\Hm\_1,s\_1) \models X \wedge \ALL\GLOBAL (q\supset \varphi\_1 \vee p) \wedge \ALL\GLOBAL(p \supset \varphi\_2)$. It is apparent that $(\Hm\_1, s\_1) \models T\_i$.

(4) For $t$=Trans(6):\\

We prove for $\EXIST\_{\tuple{ind}} \NEXT$, while for the $\ALL \NEXT$ can be proved similarly.

$(\Rto)$ For all $(\Hm\_1,s\_1) \in \Mod(T\_i)$, \ie $(\Hm\_1,s\_1) \models X \wedge \ALL\GLOBAL(q \supset \EXIST\_{\tuple{ind}}\NEXT \varphi)$\\

$\Rto$ $(\Hm\_1,s\_1) \models X$ and for all $s\_1'\in S, (\Hm\_1,s\_1') \models q \supset \EXIST\_{\tuple{ind}} \NEXT \varphi$\\

$\Rto$ $(\Hm\_1,s\_1') \models \neg q$ or there exists a state $s'$ such that $(s\_1', s') \in [ind]$ and $(\Hm\_1,s') \models \varphi$ \\

We can construct an \Ind-Kripke structure $\Hm\_2$ as follows: $\Hm\_2$ is the same with $\Hm\_1$ except for each state $s\_1'$ if $(\Hm\_1,s\_1') \models \neg q$ then $L\_2(s\_1')= L\_1(s\_1')$, else if $(\Hm\_1,s\_1') \models q$ then $L\_2(s') = L\_1(s') \cup \{p\}$. It is apparent that $(\Hm\_2,s\_1) \models \ALL\GLOBAL(q\supset \EXIST\_{\tuple{ind}} \NEXT p) \wedge \ALL\GLOBAL(p \supset \varphi)$, $(\Hm\_2,s\_2) \models T\_{i+1}$ and $(\Hm\_1, s\_1) \lrto\_{\tuple{\{p\}, {\O}}} (\Hm\_2, s\_2)$ ($s\_2=s\_1$).

$(\Lto)$ For all $(\Hm\_1, s\_1) \in \Mod(T\_{i+1})$, \ie $(\Hm\_1,s\_1) \models X \wedge \ALL\GLOBAL(q\supset \EXIST\_{\tuple{ind}} \NEXT p) \wedge \ALL\GLOBAL(p \supset \varphi)$. It is apparent that $(\Hm\_1, s\_1) \models T\_i$.

\end{proof}

\noindent\textbf{Proposition}~\ref{pro:ResE}

Let $\varphi$ be a \CTL\ formula,

%and $W$ be the set of new atoms introduced by resolution rules \textbf{(ERES1)} and \textbf{(ERES2)} (if any),

then $T\_{\varphi} \equiv\_{\tuple{V \cup V', {\O}}} Res$.

\begin{proof}(sketch)

This can be proved from $T\_i$ to $T\_{i+1}$ $(0\leq i < n)$ by using one resolution rule on $T\_i$.

By $\psi \rto\_r R\_i$ we mean using resolution rules $r$ on set $\psi$ (the formulae in $\psi$ as the premises of rule $r$) and obtaining the set $R\_i$ of resolution results.

we will show $T\_i \equiv\_{\tuple{V,I}} T\_{i+1}$ by using the resolution rule $r$. Where $T\_i= X \cup \psi$, $T\_{i+1}=X \cup R\_i$, $X$ be a set of $\CTLsnf$ clauses, $p$ be the proposition corresponding with literal $l$ used to do resolution in $r$.

(1) If $\psi \rto\_r R\_i$ by an application of $r\in \{\textbf{(SRES1)}, \dots, \textbf{(SRES8)}, (\textbf{RW1}), (\textbf{RW2})\}$, then $T\_i \equiv\_{\tuple{\{p\}, {\O}}} T\_{i+1}$.

On one hand, it is apparent that $\psi \models R\_i$ and then $T\_i \models T\_{i+1}$. On the other hand, $T\_i\subseteq T\_{i+1}$ and then $T\_{i+1} \models T\_i$.

(2) If $\psi \rto\_r R\_i$ by an application of $r=$\textbf{(ERES1)},

then $T\_i \equiv\_{\tuple{\{l, w\_{\neg l}^{\ALL}\}, {\O}}} T\_{i+1}$.

It has been proved that $\psi \models R\_i$ in~\cite{bolotov2000clausal}, then there is $T\_{i+1}=T\_i \cup \Lambda\_{\neg l}^{\ALL}$ and then for all $(\Hm\_1,s\_1) \in \Mod(T\_i= X \cup \psi)$ there is a $(\Hm\_2, s\_2)\in \Mod(T\_{i+1}=T\_i \cup \Lambda\_{\neg l}^{\ALL})$ s.t. $(\Hm\_1, s\_1) \lrto\_{\tuple{\{p, w\_{\neg l}^{\ALL}\}, {\O}}} (\Hm\_2, s\_2)$ and vice versa by Proposition~\ref{pro:TranE}.

For rule \textbf{(ERES2)} we have the same result.

\end{proof}

\noindent\textbf{Proposition}~\ref{pro:remove}

Let $V''=V \cup V'$, then we have

\[

Res \equiv\_{V''} \emph{Removing\\_atoms}(Res, V).

\]

\begin{proof}

For convenience, we let $V=\{p\}$, i.e. $V$ contain only one element $p$, $C\_i$ is a classical clause and $l$ is $p$ or $\neg p$.

It is evident that $Res \models \emph{Removing\\_atoms}(Res, V)$, hence we only need to prove that for each ${\cal K}=(\Hm, s)\in \Mod(\emph{Removing\\_atoms}(Res, V))$ with $\Hm=(S, R, L, s)$ there is an initial structure ${\cal K}'=(\Hm', s')$ such that ${\cal K} \lrto\_{V''} {\cal K}'$ and ${\cal K}' \models Res$.

As we can see that the $p$ can only occur in the right of a clause, we will prove this proposition from the following several points.

(1) We consider there are global clauses in $Res$ (the other cases are sub-cases of this one), then for each $C=\top\supset C\_1 \vee l \in Res$:

(a) If there does not exist a clause $C'\in Res$ such that $C$ and $C'$ are resolvable on $p$, this means there is no other clauses in $Res$ except $Pt$-sometime clauses $C'$ containing $\neg l$ with $Pt\in \{\ALL, \EXIST\}$.

If $p\not \in \Var(C')$, for each ${\cal K}=(\Hm, s)\in \Mod(\emph{Removing\\_atoms}(Res, V))$ we can construct $(\Hm',s')$ as follows: Let $\Hm'= (S, R, L',s)$ (i.e. $s'=s$) in which $L'$ is the same as $L$ except for each $s\_1\in S$, if $(\Hm, s\_1) \not \models C\_1 \vee l$ then let $L'(s\_1) = L(s\_1) \cup \{p\}$ if $l=p$ else $L'(s\_1) = L(s\_1) - \{p\}$.

If $C'= Q\supset Pt \FUTURE \neg l$, without loss of generality, we assume $l=p$ for each ${\cal K}=(\Hm, s)\in \Mod(\emph{Removing\\_atoms}(Res, V))$ we construct $(\Hm',s')$ as follows: let $\Hm'=(S', R', L', s')$ with $S'=S$, $R'=R$, $s'=s$ and $L'=L$ except that for each $s\in S'$ we have $L'(s) = L(s) - \{Q\}$ if $Q$ is an atom (if $Q$ is a term then we can delete the atoms which occuring in $Q$ positively and add the atoms which occuring in $Q$ negatively) and $L'(s) = L(s) \cup \{p\}$ if $(\Hm, s) \not \models C\_1$ else $L'(s) = L(s)$.

It is easy to check that ${\cal K} \lrto\_{V''} {\cal K}'$ and ${\cal K}' \models Res$.

(b) If there are some clauses $C'\in Res$ such that $C$ and $C'$ are resolvable on $p$:

\begin{enumerate}[(i)]

\item If $C'= Q\supset Pt \NEXT (C\_2 \vee \neg l)$ (we let $Pt=\GLOBAL$, we can prove similarly for $Pt = \EXIST$) then we have $Q\supset \GLOBAL \NEXT(C\_1 \vee C\_2) \in Res$, then for each ${\cal K}=(\Hm, s)\in \Mod(\emph{Removing\\_atoms}(Res, V))$ we construct $(\Hm',s')$ as follows: Let $\Hm'= (S, R, L',s)$ (i.e. $s'=s$) in which $L'$ is the same as $L$ except for each $s\_1\in S$ if $(\Hm, s\_1) \not \models Q$ then for each $(s\_1, s\_2) \in R$ if $(\Hm, s\_2) \not \models C\_1$ then let $L'(s\_2) = L(s\_2) \cup \{p\}$ if $l=p$ else $L'(s\_2) = L(s\_2) - \{p\}$, else if $(\Hm, s\_2) \models C\_1 \wedge \neg C\_2$ then let $L'(s\_2) = L(s\_2) - \{p\}$ if $l=p$ else $L'(s\_2) = L(s\_2) \cup \{p\}$; else if $(\Hm, s\_2) \models \neg C\_1 \wedge C\_2$ then let $L'(s\_2) = L(s\_2) \cup \{p\}$ if $l=p$ else $L'(s\_2) = L(s\_2) - \{p\}$. It is easy to check that ${\cal K} \lrto\_{V''} {\cal K}'$ and ${\cal K}' \models C' \wedge C$.

\item If $C' = Q\supset Pt \FUTURE \neg l$. Without loss of generality, we assume $l=p$ for convenience. In order to make $C$ and $C'$ are resolvable on $p$, there must be a set of $\CTLsnf$ clauses $\{P\_1^1 \supset \* C\_1^1$, \dots, $P\_{m\_1}^1 \supset \* C\_{m\_1}^1$, $P\_1^n \supset \* C\_1^n$, \dots, $P\_{m\_n}^1 \supset \* C\_{m\_n}^1 \}$ such that $\*$ is either empty or

an operator in $\{\GLOBAL \NEXT, \EXIST\_{\tuple{ind}} \NEXT\}$, which include $\neg C\_1 \supset l$, such that $\bigvee\_{i=1}^n \bigwedge\_{j=1}^{m\_i} P\_j^i \supset \EXIST \NEXT \EXIST \GLOBAL l$. Therefore, we get a clause $C''=\top \supset \neg Q \vee \neg p \vee C\_1$ by using ERES1 (similar for ERES2) and then $\top \supset \neg Q \vee C\_1$ by using SRES8 on $C$ and $C''$. In this case, for any ${\cal K}=(\Hm, s)\in \Mod(\emph{Removing\\_atoms}(Res, V))$ we construct $(\Hm',s')$ as follows: Let $\Hm'= (S, R, L',s)$ (i.e. $s'=s$) in which $L'$ is the same as $L$ except for each $s\_1\in S$ if $(\Hm, s\_1) \models Q$ then let $L'(s\_1) = L(s\_1) - \{p\}$, else $L'(s\_1) = L(s\_1) \cup \{p\}$. It is easy to check that ${\cal K} \lrto\_{V''} {\cal K}'$ and ${\cal K}' \models C' \wedge C$.

\item We can consider other clauses similarly, and obtained that ${\cal K} \lrto\_{V''} {\cal K}'$ and ${\cal K}' \models Res$.

\end{enumerate}

(2) We consider the $Pt$-step clauses, let $C\in Res$ is $Q \supset \GLOBAL \NEXT(C\_1 \vee \neg l)$. Without loss of generality, we assume there are some clauses $C'\in Res$ such that $C$ and $C'$ are resolvable on $p$ and $l=p$.

If $C'= Q\_1\supset Pt \NEXT (C\_2 \vee \neg l)$ (we let $Pt=\EXIST\_{ind}$, we can prove similarly for $Pt = \GLOBAL$) then we have $Q \wedge Q\_1 \supset \EXIST\_{ind} \NEXT(C\_1 \vee C\_2) \in Res$, then for each ${\cal K}=(\Hm, s)\in \Mod(\emph{Removing\\_atoms}(Res, V))$ we construct $(\Hm',s')$ as follows: Let $\Hm'= (S, R, L',s)$ (i.e. $s'=s$) in which $L'$ is the same as $L$ except for each $s\_1\in S$

\begin{enumerate}[(i)]

\item if $(\Hm, s\_1) \not \models Q \wedge Q\_1$ then ``if $(\Hm, s\_1) \models \neg Q \wedge Q\_1$ then (if $(\Hm, s\_2') \not \models C\_2$ for $(s\_1, s\_2') \in \pi\_s^{\tuple{ind}}$ then let $L'(s\_2') = L(s\_2') - \{p\}$ else $L'(s\_2') = L(s\_2')$), else if $(\Hm, s\_1) \models Q \wedge \neg Q\_1$ then for each $(s\_1, s\_2) \in R$ (if $(\Hm, s\_2) \not \models C\_1$ then let $L'(s\_2) = L(s\_2) \cup \{p\}$ else $L'(s\_2') = L(s\_2')$), else let $L'(s\_2') = L(s\_2')$".

\item else if $(\Hm, s\_1) \models Q \wedge Q\_1$ then we have $(\Hm,s\_2') \models C\_1 \vee C\_2$ for $(s\_1, s\_2) \in \pi\_s^{\tuple{ind}}$. Therefore, if $(\Hm, s\_2') \models C\_1 \wedge \neg C\_2$ then $L'(s\_2') = L(s\_2') - \{p\}$, else if $(\Hm, s\_2') \models \neg C\_1 \wedge C\_2$ then let $L'(s\_2) = L(s\_2) \cup \{p\}$ else $L'(s\_2') = L(s\_2')$. For other state $s\_2$ with $(s\_1, s\_2) \in R$ and $s\_2 \not = s\_2'$, if $(\Hm, s\_1) \models Q$ and $(\Hm, s\_2) \models \neg C\_1$ then let $L'(s\_2) = L(s\_2) \cup \{p\}$ else $L'(s\_2') = L(s\_2')$.

\end{enumerate}

It is easy to check that ${\cal K} \lrto\_{V''} {\cal K}'$ and ${\cal K}' \models C' \wedge C$, in which ${\cal K}' = (\Hm',s')$.

\end{proof}

\noindent\textbf{Proposition}~\ref{pro:Ind:EF}

Let $\EXIST\_{\tuple{ind}} \FUTURE \varphi$ be a $\CTLsnf$ formula, then we have

\[

\EXIST\_{\tuple{ind}} \FUTURE \varphi \equiv \varphi \vee \EXIST\_{\tuple{ind}} \NEXT \EXIST\_{\tuple{ind}}\FUTURE \varphi.

\]

\begin{proof}

($\Rto$) Let $(\Hm, s\_0) \in \Mod(\EXIST\_{\tuple{ind}} \FUTURE \varphi)$, then there exists a path $\pi\_{s\_0}^{\tuple{ind}}$ such that $(\Hm, s\_j) \models \varphi$ for some $s\_j \in \pi\_s^{\tuple{ind}}$ with $0 \leq j$. In this case, we can see either $j=0$ or $j > 0$, then we have $(\Hm, s\_0) \models \varphi \vee \EXIST\_{\tuple{ind}} \NEXT \EXIST\_{\tuple{ind}}\FUTURE \varphi$.

($\Lto$) Let $(\Hm, s\_0) \in \Mod(\varphi \vee \EXIST\_{\tuple{ind}} \NEXT \EXIST\_{\tuple{ind}}\FUTURE \varphi)$, then we have $(\Hm,s\_0) \models \varphi$ or there exists a path $\pi\_{s\_0}^{\tuple{ind}} = (s\_0, s\_1, \dots)$ such that $(\Hm, s\_1) \models \EXIST\_{\tuple{ind}}\FUTURE \varphi$. Therefore, we have $(\Hm, s\_0) \models \EXIST\_{\tuple{ind}} \FUTURE \varphi$ by the semantic of $\EXIST\_{\tuple{ind}}\FUTURE$.

\end{proof}

\noindent\textbf{Proposition}~\ref{pro:In2NI}

Let $P$, $P\_i$ and $\varphi\_i$ be \CTL\ formulas, then

\begin{enumerate}[(i)]

\item $\bigwedge\_{i=1}^n (P\supset \EXIST\_{\tuple{ind}} \NEXT \varphi\_i) \equiv\_{\tuple{\emptyset, \{ind\}}} P\supset \EXIST \NEXT \bigwedge\_{i=1}^n \varphi\_i$,

\item $\bigwedge\_{i=1}^n (P\_i\supset \EXIST\_{\tuple{ind}} \NEXT \varphi\_i) \equiv\_{\tuple{\emptyset, \{ind\}}} \bigwedge\_{e \in 2^{\{0,\dots, n\}} \setminus \{\emptyset\}}(\bigwedge\_{i\in e}P\_i\supset \EXIST \NEXT (\bigwedge\_{i\in e}\varphi\_i))$,

\item $\bigwedge\_{i=1}^n (P\supset \EXIST\_{\tuple{ind}} \FUTURE \varphi\_i) \equiv\_{\tuple{\emptyset, \{ind\}}} P\supset \bigvee\EXIST\FUTURE (\varphi\_{j\_1} \wedge \EXIST\FUTURE(\varphi\_{j\_2} \wedge \EXIST\FUTURE(\dots \wedge \EXIST\FUTURE \varphi\_{j\_n})))$, where $(j\_1, \dots, j\_n)$ are sequences of all elements in $\{0, \dots, n\}$,

\item $P\supset (C \vee \EXIST\_{\tuple{ind}} \NEXT \varphi\_1) \wedge P \supset \EXIST\_{\tuple{ind}} \NEXT \varphi\_2 \equiv\_{\tuple{\emptyset, \{ind\}}} P \supset ((C \wedge \EXIST \NEXT \varphi\_2) \vee \EXIST \NEXT (\varphi\_1 \wedge \varphi\_2))$,

\item $P\supset (C \vee \EXIST\_{\tuple{ind}} \NEXT \varphi\_1) \vee P \supset \EXIST\_{\tuple{ind}} \NEXT \varphi\_2 \equiv\_{\tuple{\emptyset, \{ind\}}} P \supset (C \vee \EXIST \NEXT (\varphi\_1 \vee \varphi\_2))$.

\end{enumerate}

\begin{proof}

(i) For all $(\Hm, s\_0) \in \Mod(\bigwedge\_{i=1}^n (P\supset \EXIST\_{\tuple{ind}} \NEXT \varphi\_i))$ there exists $(s\_0, s\_1)\in [ind]$ such that $(\Hm, s\_1) \models \varphi\_1$, \dots, $(\Hm, s\_1) \models \varphi\_n$, then there is $(s\_0, s\_1)\in R$ s.t. $(\Hm, s\_1) \models \bigwedge\_{i=1}^n \varphi\_i$, i.e. $(\Hm, s\_0) \models P\supset \EXIST \NEXT \bigwedge\_{i=1}^n \varphi\_i$.

For each $(\Hm, s\_0) \in \Mod(P\supset \EXIST \NEXT \bigwedge\_{i=1}^n \varphi\_i)$, we suppose there is $(s\_0, s\_1)\in R$ s.t. $(\Hm, s\_1) \models \bigwedge\_{i=1}^n \varphi\_i$. It is easy to construct an initial \Ind-model $(\Hm', s\_0)$ such that $(\Hm', s\_0)$ is identical to $(\Hm, s\_0)$ except the $(s\_0, s\_1) \in [ind]$, i.e. $(\Hm, s\_0) \lrto\_{\tuple{\emptyset, \{ind\}}} (\Hm', s\_0)$.

(ii) (If part) For any model $(\Hm,s\_0)$ of the left side of the equation if there is $(\Hm,s\_0) \models \bigwedge\_{i=1}^m P\_{j\_i}$ with $j\_i \in \{1, \dots, n\}$ and $1\leq m \leq n$, then there is a next state $s\_1$ of $s\_0$ with $(s\_0, s\_1) \in [ind]$ such that $(\Hm, s\_1) \models \bigwedge\_{i=1}^m \varphi\_{j\_i}$. By the definition of $[ind]$, we have $(s\_0, s\_1) \in R$ and then $(\Hm, s\_0) \models \bigwedge\_{i=1}^m P\_{j\_i} \supset \EXIST \NEXT (\bigwedge\_{i=1}^m P\_{j\_i} \varphi\_{j\_i})$. The other side can be similarly proved as (i).

(iii) (Only if part) For any model $(\Hm,s\_0)$ of the right side of the equation if there is $(\Hm,s\_0) \models P$ then there exists a path $\pi\_{s\_0}$ such that $\varphi\_i \in \pi\_{s\_0}$ ($1\leq i \leq n$). This means we can construct an initial \Ind-model $(\Hm', s\_0)$ such that $(\Hm', s\_0)$ is identical to $(\Hm, s\_0)$ except for each $(s\_j, s\_{j+1})$ of $\pi\_{s\_0}$ there is $(s\_j, s\_{j+1}) \in [ind]$ $(0\leq j)$. It is easy to check $(\Hm', s\_0) \models \bigwedge\_{i=1}^n (P\supset \EXIST\_{\tuple{ind}} \FUTURE \varphi\_i)$ and $(\Hm, s\_0) \lrto\_{\tuple{\emptyset, \{ind\}}} (\Hm', s\_0)$. The other side can be shown similarly as in (ii).

Other results can be proved similarly.

\end{proof}

\noindent\textbf{Proposition}~\ref{pro:complexity}

Let $\varphi$ be a CTL formula and $V \subseteq \Ha$.

The time and space complexity of Algorithm~\ref{alg:compute:forgetting:by:Resolution} are $O((m+1)2^{4(n+n')}$. Where $|\Var(\varphi)|=n$, $|V'|=n'$ ($V'$ is set of atoms introduced in the Transform process) and $m$ is the number of indices introduced during transformation.

\begin{proof}

It is mainly decided by the resolution process.

The possible number of $\CTLsnf$ clauses under the give $V$, $V'$ and $Ind$ is $(m+1)2^{4(n+n')}+(m\*(n+n')+n+n'+1)2^{2(n+n')+1})$.

\end{proof}

\noindent\textbf{Theorem}~\ref{thm:Aclm}[Generalised Ackermann’s Lemma]

Let $\Gamma$ be a set of formulae that contains the set $\Delta = \{\top \supset \neg x \vee C\_1$, \dots, $\top \supset \neg x \vee C\_n, x \supset B\_1, \dots, x \supset B\_m\}$ of clauses, where $x \in V'$ is an atom introduced in the transformation process, the $C\_i$ $(1 \leq i \leq n)$ are classical propositional clauses that do not contain $x$, and $B\_j$ ($1 \leq j \leq m$) are formulae of the disjunction (or conjunction) of formulae of form $Qt {\cal T} C$ with $Qt$ is empty (with ${\cal T}$ is empty) or $Qt \in \{\ALL, \EXIST\}$, ${\cal T}\in \{\NEXT, \FUTURE\}$ and $C$ is a CNF (or DNF) that also do not contain $x$. If $\Gamma'= \Gamma - \Delta$ is positive w.r.t. $x$ (i.e. each clause in $\Gamma'$ is positive w.r.t. $x$), then $\Gamma'[x/\varphi] \equiv\_{\tuple{\{x\}, \emptyset}} \Gamma$ with $\varphi = \bigwedge\_{i=1}^n C\_i \wedge \bigwedge\_{j=1}^m B\_j$, where $\Gamma'[x/\varphi]$ is obtained from $\Gamma'$ by replacing all $x$ with $\varphi$.

\begin{proof}

Without loss of generality, we suppose there are only A-step clauses in $\Gamma'$, other cases can be proved similarly.

$(\Rto)$ For any model $(\Hm, s\_0)$ of $\Gamma$, it is obvious that $(\Hm, s\_0) \models \Gamma'$.

$(\Lto)$ For any models $(\Hm, s\_0)$ of $\Gamma'$ with $\Hm = (S, R, L, s\_0)$, we can construct an \Ind-initial structure $\Hm'=(S', R', L', s\_0')$ with $S'=S$, $R'=R$, $s\_0'= s\_0$ and $L'$ is the same with $L$ except that for each $s'\in S'$ if $(\Hm', s') \models x \wedge \vaprhi$ then let $L'(s') = L(s) - \{x\}$, else let $L'(s') = L(s)$.

It is easy to check that $(\Hm,s\_0) \lrto\_{\tuple{\{x\}, \emptyset}} (\Hm', s\_0')$ and $(\Hm',s\_0') \models \Gamma'$.

\end{proof}

\end{document}