Natural deduction calculus for computation tree logic.

Alexander Bolotov[1]
Oleg Grigoriev[2]
Vasilyi Shangin[2]

[1] Harrow School of Computer Science, University of Westminster
[2] Department of Logic, Faculty of Philosophy, Moscow State University

Natural Deduction Calculus for Computation Tree Logic

Alexander Bolotov
Harrow School of Computer Science, University of Westminster
Watford Road, Harrow HA1 3TP, UK.
A.Bolotov@wmin.ac.uk

Oleg Grigoriev, Vasilyi Shangin
Department of Logic, Faculty of Philosophy
Moscow State University, Moscow, 119899, Russia.
{shangin,grig}@philos.msu.ru

Abstract

We present a natural deduction calculus for the Computation Tree Logic, CTL, defined with the full set of classical and temporal logic operators. The system extends the natural deduction construction of the linear-time temporal logic. This opens the prospect to apply our technique as an automatic reasoning tool in a deliberative decision making framework across various applications in AI and Computer Science, where the branching-time setting is required.

1 Introduction

This work on natural deduction proof system for branching-time logic has been carried out as part of our investigation of various topics of use of formal specification and deductive reasoning techniques related to the following problem structure. Our interest lies in the area of complex information systems that we have proposed to model in a generic multi-layer architecture [1]. Among these layers we distinguish the functionality layer ('FL' for short), the management layer ('ML') and the reasoning layer ('RL'). These layers represent respectively the functionality of the system, the configuration and re-configuration management, and, finally, the automated reasoning engine. Within each of the layers we embed a relevant model.

Within the FL the model represents the main functionality of the system, which consists of functional components that carry out the required processing tasks. The management layer manages the configuration of these 'components' by providing the means to monitor and reconfigure the functional components. Finally, the reasoning layer communicates with the management layer in order to determine which reconfigurations are plausible and propose reconfigurations to the management layer. Our main focus has been on the reasoning layer. Due to the dynamic and non-deterministic nature of the underlying complex systems, we believe that the appropriate formal framework is given in the so called branching-time setting. In [1] we have shown how the specification language of the normal form for branching-time logic [3] can be used in this framework to enable a resolution based deductive verification.

The other set of problems relevant to the temporal reasoning within our problem structure is to equip the reasoning layer with the relevant goal-directed deductive technique to enable its problem-solving. Looking for the corresponding deductive reasoning techniques, we have become interested in natural deduction constructions for temporal logic. The definition of natural deduction proof technique for non-classical logic in general, and for temporal logic, in particular, is also an important and challenging theoretical task.

The particular approach to build an ND-calculus we are interested in is described in detail in [4]. It is a modification of Quine's representation of subordinate proof [11] developed for classical propositional and first-order logic. The ND technique initially defined for classical propositional logic was extended to first-order logic [4, 5]. It has also been extended to the non-classical framework of propositional intuitionistic logic [10].

In [2] we have developed an ND system for propositional linear-time temporal logic. In this paper we extend this approach to capture the Computation Tree Logic (CTL) [6] as the most commonly used logic for the desired branching-time setting.

The paper is organized as follows. In §2 we review the syntax and semantics of CTL. In §3 we describe and give examples of the Natural Deduction System for CTL henceforth referred to as CTL_{ND} and then in §4 present the correctness argument. Finally, in §5, we provide concluding remarks and identify future work.

2 Syntax and Semantics of CTL

2.1 CTL Syntax

We define the language of the computation tree logic (CTL) using the following symbols.

- a set, $Prop$, of atomic propositions: $p, q, r, \ldots, p_1, q_1, r_1, \ldots, p_n, q_n, r_n, \ldots$;
- classical operators: $\neg, \wedge, \Rightarrow, \vee$;
- temporal operators:

$\Box$	–	'always in the future';
$\Diamond$	–	'at sometime in the future';
$\bigcirc$	–	'at the next moment in time';
$\mathcal{U}$	–	'until'.

- path quantifiers:

A	–	'for any future path;
E	–	'for some future path.

In the syntax of CTL we distinguish *state* (S) and *path* (P) formulae, such that well formed formulae are state formulae. These classes of formulae are inductively defined below (where C is a formula of classical propositional logic)

$$S ::= C|S \wedge S|S \vee S|S \Rightarrow S|\neg S|\mathbf{A}P|\mathbf{E}P$$
$$P ::= \Box S|\Diamond S|\bigcirc S|S\,\mathcal{U}\,S$$

Recall that the distinguished feature of CTL formulae is that any temporal operator must be immediately preceded by a path quantifier. Thus, examples of CTL formulae are $\mathbf{A}\bigcirc\mathbf{E}(p\,\mathcal{U}\,q), \mathbf{A}(\mathbf{E}(p\,\mathcal{U}\,q)\,\mathcal{U}\,(\mathbf{A}\bigcirc q))$.

Note that $\mathcal{U}$ and $\bigcirc$ form a functionally complete set of temporal operators in the propositional linear-time temporal logic and thus the other standard temporal operators, $\Box$ and $\Diamond$ are expressible via this set [9]. However, we here consider the formulation of CTL with the full set of classical and temporal operators taking into account the use of the logic in the specification of complex dynamic distributed systems.

A temporal operator paired with a path quantifier is called the *basic modality* of CTL. We will essentially use the concept of basic modality in our construction of the natural deduction system for CTL, namely, our rules will be applied to some basic modality **PT** where **P** is either of the path quantifiers and **T** is either of the temporal operators.

2.2 CTL Semantics

We first introduce the notation of tree structures, the underlying structures of time assumed for branching-time logics, which we utilise in our presentation.

Definition 1 *A* tree *is a pair (S, R), where S is a set of states and $R \subseteq S \times S$ is a relation between states of S such that*

- *$s_0 \in S$ is a unique root node, i.e. there is no state $s_i \in S$ such that $R(s_i, s_0)$;*
- *for every $s_i \in S$ there exists $s_j \in S$ such that $R(s_i, s_j)$;*
- *for every s_i, s_j, $s_k \in S$, if $R(s_i, s_k)$ and $R(s_j, s_k)$ then $s_i = s_j$.*

A *path*, χ_{s_i} is a sequence of states $s_i, s_{i+1}, s_{i+2} \ldots$ such that for all $j \geq i$, $(s_j, s_{j+1}) \in R$. Let $\boldsymbol{\chi}$ be a family of all paths of $\mathcal{M}$. A path $\chi_{s_0} \in \boldsymbol{\chi}$ is called a *fullpath*. Let X be a family of all fullpaths of $\mathcal{M}$. Given a path χ_{s_i} and a state $s_j \in \chi_{s_i}$, $(i < j)$ we term a finite subsequence $[s_i, s_j] = s_i, s_{i+1}, \ldots, s_j$ of χ_{s_i} *a prefix* of a path χ_{s_i} and an infinite sub-sequence $s_j, s_{j+1}, s_{j+2} \ldots$ of χ_{s_i} *a suffix* of a path χ_{s_i} abbreviated $Suf(\chi_{s_i}, s_j)$.

Definition 2 (Branching degree of a state) *The number of immediate successors of a state $s_i \in S$ in a tree (S, R) is called the* branching degree *of s_i.*

In a general case a state of a tree can have an infinite number of successors. However, following [8] (page 1011), trees with arbitrary, even uncountable, branching, "as far as our branching temporal logic are concerned, are indistinguishable from trees with finite, even bounded, branching". Thus, without loss of generality, we assume that underlying CTL tree models are of at most countable branching.

Definition 3 (Total countable ω-tree) *A countable ω-tree, τ_ω, is a tree (S, R) with the family of all fullpaths, X, which satisfies the following conditions:*

- *each fullpath is isomorphic to natural numbers;*
- *every state $s_m \in S$ has a countable number of successors;*
- *X is R-generable* [8], *i.e. for every state $s_m \in S$, there exists $\chi_n \in X$ such that $s_m \in \chi_n$, and for every sequence $\chi_n = s_0, s_1, s_2 \ldots$ the following is true: $\chi_n \in X$ if, and only if, for every m $(1 \leq m)$, $R(s_m, s_{m+1})$.*

Since in ω trees fullpaths are isomorphic to natural numbers, in the rest of the paper we will abbreviate the relation R as $\leq$.

We interpret a well-formed CTL formula in a structure $\mathcal{M} = \langle S, \leq, s_0, X, L\rangle$, where $(S, \leq)$ is a countable ω-tree with a root s_0, X is a set of all fullpaths and L is an interpretation function mapping atomic propositional symbols to truth values at each state.

Recall that since the underlying CTL structures are R-generable, they are suffix, fusion and limit closed [8].

Now in Figure 1 we define a relation '$\models$', which evaluates well-formed CTL formulae at a state s_m in a model $\mathcal{M}$ (in the rest of the paper we will use "iff" to abbreviate the expression "if, and only, of").

$\langle\mathcal{M}, s_m\rangle \models p$	iff	$p \in L(s_m)$, for $p \in Prop$.
$\langle\mathcal{M}, s_m\rangle \models \neg A$	iff	$\langle\mathcal{M}, s_m\rangle \not\models A$
$\langle\mathcal{M}, s_m\rangle \models A \wedge B$	iff	$\langle\mathcal{M}, s_m\rangle \models A$ and $\langle\mathcal{M}, s_m\rangle \models B$
$\langle\mathcal{M}, s_m\rangle \models A \vee B$	iff	$\langle\mathcal{M}, s_m\rangle \models A$ or $\langle\mathcal{M}, s_m\rangle \models B$
$\langle\mathcal{M}, s_m\rangle \models A \Rightarrow B$	iff	$\langle\mathcal{M}, s_m\rangle \not\models A$ or $\langle\mathcal{M}, s_m\rangle \models B$
$\langle\mathcal{M}, s_m\rangle \models \mathbf{A}B$	iff	for each χ_{s_m}, $\langle\mathcal{M}, \chi_{s_m}\rangle \models B$
$\langle\mathcal{M}, s_m\rangle \models \mathbf{E}B$	iff	there exists χ_{s_m} such that $\langle\mathcal{M}, \chi_{s_m}\rangle \models B$
$\langle\mathcal{M}, \chi_{s_m}\rangle \models A$	iff	$\langle\mathcal{M}, s_m\rangle \models A$, for state formula A
$\langle\mathcal{M}, \chi_{s_m}\rangle \models \Box B$	iff	for each $s_n \in \chi_{s_m}$, if $m \leq n$ then $\langle\mathcal{M}, Suf(\chi_{s_m}, s_n)\rangle \models B$
$\langle\mathcal{M}, \chi_{s_m}\rangle \models \Diamond B$	iff	there exists $s_n \in \chi_{s_m}$ such that $m \leq n$ and $\langle\mathcal{M}, Suf(\chi_{s_m}, s_n)\rangle \models B$
$\langle\mathcal{M}, \chi_{s_m}\rangle \models \bigcirc B$	iff	$\langle\mathcal{M}, Suf(\chi_{s_m}, s_{m+1})\rangle \models B$
$\langle\mathcal{M}, \chi_{s_m}\rangle \models A\,\mathcal{U}\,B$	iff	there exists $s_n \in \chi_{s_m}$ such that $m \leq n$ and $\langle\mathcal{M}, Suf(\chi_{s_m}, s_n)\rangle \models B$ and for each $s_k \in \chi_{s_m}$, if $m \leq k < n$ then $\langle\mathcal{M}, Suf(\chi_{s_m}, s_k)\rangle \models A$

Figure 1. CTL semantics

Definition 4 *[Satisfiability] A well-formed CTL formula, B, is satisfiable if, and only if, there exists a model $\mathcal{M}$ such that $\langle\mathcal{M}, s_0\rangle \models B$.*

Definition 5 *[Validity] A well-formed CTL formula, B, is valid if, and only if, it is satisfied in every possible model.*

3 Natural Deduction System $\mathrm{CTL_{ND}}$

3.1 Extended CTL Syntax and Semantics

To define the rules of the natural system we extend the syntax of CTL by introducing labelled formulae.

Firstly, we define the set of labels,

$$Lab = Lab_S \cup Lab_P$$

where

$$Lab_S = \{x, y, z, \dots\}$$

is a set of variables interpreted over states of a tree and

$$Lab_P = \{\alpha, \beta, \gamma, \dots\}$$

is a set of variables over paths, elements of a tree.

We will distinguish universal and rigid variables. This second type of variables is linked with the restrictions on the application of some of the rules which will be explained later. In the rest of the paper we will refer to the sets of labels that represent universal and rigid variables as to Lab_S^{univ}, Lab_P^{univ} and Lab_S^{rigid}, Lab_P^{rigid}, respectively.

We then define two binary relations '$\preceq$' and 'Next', and the operation $'$ using the following notation. By $(i \preceq j)_\varphi$ or $Next(i, j)_\varphi$ we abbreviate "$i \preceq j$ (or $Next(i, j)$) holds in an (arbitrary or some) branch φ (depending on whether φ is universal or rigid), which starts at a point i and includes j", i.e. we agree that the starting point of path φ is the state that corresponds to the first state variable, i, in the relation $\preceq$ or $Next$.

Definition 6 (Relations $\prec, \simeq, \preceq$ and $Next$, operation $'$) *Given a countable total ω-tree, $(S, \leq)$, with the set of paths, $\boldsymbol{\chi}$, let g_S be a function from Lab_S to S, and g_P be a function from Lab_P to $\boldsymbol{\chi}$. Then for any i, $j \in Lab_S$ and $\varphi \in Lab_P$:*

(6.1) *$g_S(i)$ is the least element in $g_P(\varphi)$* iff *if $g_S(j) \in g_P(\varphi)$ and $g_S(j) < g_S(i)$ then $g_S(i) = g_S(j)$,*

(6.2) $(i \prec j)_\varphi$ iff *if $g_S(i)$ and $g_S(j)$ are in $g_P(\varphi)$ then $g_S(i) < g_S(j)$ and $g_S(i)$ is the least element in $g_P(\varphi)$,*

(6.3) $(i \simeq j)_\varphi$ iff *if $g_S(i)$ and $g_S(j)$ are in $g_P(\varphi)$ then $g(i) = g(j)$,*

(6.4) $(i \preceq j)_\varphi$ iff *if $i \prec j$ or $i \simeq j$,*

(6.5) $(i \preceq j)_\varphi$ iff *if $g_S(i)$ and $g_S(j)$ are in $g_P(\varphi)$ then $g_S(i) < g_S(j)$ and $g_S(i)$ is least in $g_P(\varphi)$ or $g_S(i) = g_S(j)$ and $g_S(i)$ is the least element in $g_P(\varphi)$,*

(6.6) $Next(i, j)_\varphi$ iff *if $g_S(i)$ and $g_S(j)$ are in $g_P(\varphi)$ then $g(j) = g(i) + 1$ and $g_S(i)$ is the least element in $g_P(\varphi)$,*

i.e. 'Next' is the 'predecessor-successor' relation such that for any $i \in Lab_S$, there exists $j \in Lab_S$ such that $Next(i, j)_\varphi$ (seriality),

(6.7) *Given a label i, the operation $'$ applied to i gives us the label i' such that $Next(i, i')_\varphi$.*

We also introduce the notation $(i \preceq j)_{sf\alpha_k}$ to abbreviate that $i \preceq j$ holds in $Suf(\alpha_k, i)$ for an arbitrary (or some) branch α_k.

The following properties follow straightforwardly from Definition 6.

For any $i, j, k \in Lab_S$ and $\varphi, \psi \in Lab_P$

- if $(i \prec j)_\varphi$ then $(i \preceq j)_\varphi$,
- if $Next(i,j)_\varphi$ then $(i \preceq j)_\varphi$,
- $(i \preceq i)_\varphi$ (reflexivity),
- if $(i \preceq j)_\varphi$ and $(j \preceq k)_\psi$ then $(i \preceq k)_{\psi'}$, where
  - ψ' is a new label from Lab_P^{rigid} if $\varphi \in Lab_P^{rigid}$ or $\varphi \in Lab_P^{rigid}$ (transitivity);
  - otherwise, $\psi' \in Lab_P^{univ}$,
- if $(i \preceq j)_{sf\varphi}$ and $(j \preceq k)_{sf\varphi}$ then $(i \preceq k)_{sf\varphi}$,
- if $(i \preceq i')_{sf\varphi}$ and $Next(i'i'')_{sf\varphi'_i}$ then $(i \preceq i'')_{sf\varphi}$.

Following [12], the expressions representing the properties of $\preceq$ and $Next$ are called 'relational judgements'.

Now we are ready to introduce the CTL_{ND} syntax.

Definition 7 ($\mathbf{CTL_{ND}}$ Syntax)

- *If A is a CTL formula and $i \in Lab_S$ then $i : A$ is a CTL_{ND} formula.*
- *Any relational judgement of the type $Next(i,i')_\varphi$, $Next(i,i')_{sf\varphi_k}$, $(i \preceq j)_\varphi$, and $(i \preceq j)_{sf\varphi}$, where $i, j \in Lab_S$ and $\varphi \in Lab_P$ is a CTL_{ND} formula.*

$\mathbf{CTL_{ND}}$ Semantics. For the interpretation of CTL_{ND} formulae we adapt the semantical constructions previously defined for the logic CTL. In the rest of the paper we will use capital letters $A, B, C, D, \ldots$ as metasymbols for CTL formulae, and calligraphic letters $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D} \ldots$ to abbreviate formulae of CTL_{ND}, i.e. either labelled formulae or relational judgements. The intuitive meaning of $i : A$ is that A is satisfied at the world $i \in Lab_S$. Thus, based on our observations above, we simply need the following statements.

Let Γ be a non-empty set of CTL_{ND} formulae, let $Lab_S^\Gamma = \{x | x : A \in \Gamma\}$, $Lab_P^\Gamma = \{\varphi | \mathcal{B}_\varphi \in \Gamma\}$ (where $\mathcal{B}$ abbreviates relational judgements), let $\mathcal{M} = \langle S, R, s_0, X, L\rangle$ be a model as defined in §2, and let $\boldsymbol{\chi}$ be a set of paths. Note that obviously since $Lab_S^\Gamma \subseteq Lab_S$ and $Lab_P^\Gamma \subseteq Lab_P$ functions g_S and g_P introduced by Definition 6 are also defined for these sets. Let us abbreviate these specific cases of the mappings from Lab_S^Γ to S and from Lab_P^Γ to $\boldsymbol{\chi}$ as g_S^Γ and g_P^Γ, respectively. Moreover, if Γ contains a relational judgement $\mathcal{B}_\varphi$ then, depending on its structure, items (6.1) - (6.6) of Definition 6 are satisfied.

In the following definition we introduce a notion of realisability of Γ in a model $\mathcal{M}$ utilising Definition 6. Since Γ can contain formulae of the type $i : A$ or relational judgements we will tackle each of these cases, treating possible variations of world and path labels.

Definition 8 (Realisation of $\mathbf{CTL_{ND}}$ formulae in a model) *Model $\mathcal{M}$* realises *a set of CTL_{ND} formulae, Γ, if the following conditions hold:*

(1) For any $i \in Lab_S^\Gamma$, such that $i \in Lab_S^{univ}$ and for any A, if $i : A \in \Gamma$ then $\langle \mathcal{M}, g_S^\Gamma(i)\rangle \models A$ for any function g_S^Γ.

(2) For any $i \in Lab_S^\Gamma$, such that $i \in Lab_S^{rigid}$ and for any A, if $i : A \in \Gamma$ then $g_S'^\Gamma(i) = g_S^\Gamma(i)$ and $\langle \mathcal{M}, g_S^\Gamma(i)\rangle \models A$ for any function g_S^Γ and $g_S'^\Gamma$.

(3) For any i, j, k, φ, if $(i \prec j)_\varphi \in \Gamma$, $[(i \prec j)_{sf\varphi}$, $(i \preceq j)_\varphi$, $(i \preceq j)_{sf\varphi}$, $Next(i,i')_\varphi$, $Next(i,i')_{sf\varphi} \in \Gamma]$ then the relation $\prec_\varphi$ $[\prec_{sf\varphi}$, $\preceq_\varphi$, $\preceq_{sf\varphi}$, $Next_\varphi$, $Next_{sf\varphi}]$ is defined

- for all g_S^Γ, if g_S^Γ is applied to $k \in Lab_S^{univ}$,

- for all g_S^Γ, if g_S^Γ is applied to $k \in Lab_S^{rigid}$ and $g_S^\Gamma(k) = g_S'^\Gamma(k)$, for any g_S^Γ and $g_S'^\Gamma$,

for the tree on which the model $\mathcal{M}$ is based.

The set Γ in this case is called realisable.

Definition 9 ($\mathbf{CTL_{ND}}$ Validity) *A well-formed CTL_{ND} formula, $\mathcal{A} = i : B$, is valid (abbreviated as $\models_{ND} \mathcal{A}$) if, and only if, the set $\{\mathcal{A}\}$ is realisable in every possible model, for any function f.*

3.2 Rules for Boolean Operations

The set of rules is divided into the two classes: *elimination* and *introduction* rules. Rules of the first group allow us to simplify formulae to which they are applied. These are rules for the 'elimination' of logical constants. Rules of the second group are aimed at 'building' formulae, introducing new logical constants.

Below we define the sets of elimination and introduction rules, where '*el*' and '*in*' that follow a Boolean operation abbreviate an elimination or an introduction rule of this operation.

	Elimination Rules :		Introduction Rules :
$\wedge\, el_1$	$\dfrac{i : A \wedge B}{i : A}$	$\wedge\, in$	$\dfrac{i : A \quad i : B}{i : A \wedge B}$
$\wedge\, el_2$	$\dfrac{i : A \wedge B}{i : B}$	$\vee\, in_1$	$\dfrac{i : A}{i : A \vee B}$
$\vee\, el$	$\dfrac{i : A \vee B \quad i : \neg A}{i : B}$	$\vee\, in_2$	$\dfrac{i : B}{i : A \vee B}$
$\Rightarrow el$	$\dfrac{i : A \Rightarrow B \quad i : A}{i : B}$	$\Rightarrow in$	$\dfrac{[i : C] \quad i : B}{i : C \Rightarrow B}$
$\neg\, el$	$\dfrac{i : \neg\neg A}{i : A}$	$\neg\, in$	$\dfrac{[j : C] \;\; i : B \;\; i : \neg B}{j : \neg C}$

In the formulation of the rules '$\Rightarrow in$' and '$\neg\ in$' formulae $[i : C]$ and $[j : C]$ respectively must be the most recent non discarded [5] assumptions occurring in the proof. When we apply one of these rules on step n and discard an assumption on step m, we also discard all formulae from m to $n-1$. We will write $[m-(n–1)]$ to indicate this situation.

3.3 Rules for Temporal Logic

Elimination Rules:

$$\mathbf{A}\bigcirc el \quad \frac{i : \mathbf{A}\bigcirc A}{Next(i,i')_\varphi, \quad i' : A} \quad i' : A \in M1$$

$$\mathbf{E}\bigcirc el \quad \frac{i : \mathbf{E}\bigcirc A}{Next(i,i')_\varphi, \quad i' : A} \quad \begin{array}{c}\varphi \in Lab_P^{rigid}, \\ i' : A \in M1\end{array}$$

$$\mathbf{A}\Box el \quad \frac{i : \mathbf{A}\Box A}{(i,j)_\varphi, \quad j : A}$$

$$\mathbf{E}\Box el \quad \frac{i : \mathbf{E}\Box p}{(i,j)_\varphi, \quad j : A} \quad \varphi \in Lab_p^{rigid}$$

$$\mathbf{E}\Diamond el \quad \frac{i : \mathbf{E}\Diamond A}{(i,j)_\varphi, \quad j : A} \quad \begin{array}{c} j \in Lab_s^{rigid}, \ \varphi \in Lab_p^{rigid} \\ j \mapsto i, \ \forall C(j : C \notin M1)\end{array}$$

$$\mathbf{A}\Diamond el \quad \frac{i : \mathbf{A}\Diamond A}{(i,j)_\varphi, \quad j : A} \quad \begin{array}{c} j \in Lab_s^{rigid}, j \mapsto i \\ \forall C(j : C \notin M1)\end{array}$$

$$\mathbf{E}\mathcal{U}\, el_1 \frac{i : \mathbf{E}(A \mathcal{U} B), \quad i : \neg B}{i : A, \ (i,j)_\varphi, \ j : B}$$

$\varphi \in Lab_P^{rigid}, \ j \in Lab_S^{rigid}, \ j \mapsto i, \ \forall C(j : C \notin M1)$

$$\mathbf{A}\mathcal{U}\, el_1 \frac{i : \mathbf{A}(A \mathcal{U} B), \quad i : \neg B}{i : A, \ (i \preceq j)_\varphi, \ j : B} \quad \begin{array}{c} j \in Lab_s^{rigid}, \ j \mapsto i \\ \forall C(j : C \notin M1)\end{array}$$

$\mathbf{P}\mathcal{U}\, el_2$

$$\frac{(i^{[AB]} \preceq j^{[AB]})_\varphi, (i^{[AB]} \preceq k)_\varphi, (k \prec j^{[AB]})_\varphi}{k : A}$$

Introduction Rules:

$$\mathbf{E}\bigcirc in \quad \frac{i' : A, \quad Next(i,i')_\varphi}{i : \mathbf{E}\bigcirc A}$$

$$\mathbf{A}\bigcirc in \quad \frac{i' : A, \quad Next(i,i')_\varphi}{i : \mathbf{A}\bigcirc A} \quad \begin{array}{c}\varphi \notin Lab_P^{rigid} \\ i' \notin Lab_S^{rigid}\end{array}$$

$$\mathbf{A}\Box in \quad \frac{j : A, \quad [(i \preceq j)_\varphi]*}{i : \mathbf{A}\Box A} \quad \begin{array}{c}\varphi \notin Lab_p^{rigid} \\ j \notin Lab_s^{rigid} \\ j : A \notin M1\end{array}$$

$$\mathbf{E}\Box in \quad \frac{j : A, \quad [(i \preceq j)_\varphi]*}{i : \mathbf{E}\Box A} \quad \begin{array}{c} j \notin Lab_s^{rigid} \\ j : A \notin M1\end{array}$$

$$\mathbf{A}\Diamond in \quad \frac{j : A, \quad (i \preceq j)_\varphi}{i : \mathbf{A}\Diamond A} \quad \varphi \notin Lab_p^{rigid}$$

$$\mathbf{E}\Diamond in \quad \frac{j : A, \quad (i \preceq j)_\varphi}{i : \mathbf{E}\Diamond A}$$

$$\mathbf{E}\mathcal{U}\, in_1 \frac{i : A, \quad i' : B, \quad Next(i,i')_\varphi}{i : \mathbf{E}(A \mathcal{U} B)}$$

$\mathbf{E}\mathcal{U}\, in_2$

$$\frac{j : A, \ l : B, \ (i \preceq l)_{sf\varphi}, \ [(i \preceq j)_{sf\varphi}]*, \ [(j \preceq l)_{sf\varphi}]*}{i : \mathbf{E}(A \mathcal{U} B)}$$

$where \ j \notin Lab_S^{rigid}, j : A \notin M1$

$$\mathbf{A}\mathcal{U}\, in_1 \frac{i : B}{i : \mathbf{A}(A \mathcal{U} B)}$$

$$\mathbf{A}\mathcal{U}\, in_2 \frac{i : A, \quad i' : B, \quad Next(i,i')_\varphi}{i : \mathbf{A}(A \mathcal{U} B)}$$

$\varphi \notin Lab_P^{rigid}$

$\mathbf{A}\mathcal{U}\, in_3$

$$\frac{j : A, \ l : B, \ (i \preceq l)_{sf\varphi}, \ [(i \preceq j)_{sf\varphi}]*, \ [(j \preceq l)_{sf\varphi}]*}{i : \mathbf{A}(A \mathcal{U} B)}$$

$\varphi \notin Lab_P^{rigid}, \ j \notin \ Lab_S^{rigid}, \ j : A \notin M1$

- If a type of a variable that occurs in a premise of a rule is not indicated then it can be either universal or rigid.
- The condition $\forall C(j : C \notin M1)$ in the rules $\mathbf{P}\Diamond el$ and $\mathbf{P}\mathcal{U}\, el_1$ means that the label j should not occur in the proof in any formula, C, that is marked by $M1$.
- The condition $j : A \notin M1$ in the rules $\mathbf{P}\Box in$ and $\mathbf{P}\mathcal{U}\, in_3$ means that $j : A$ is not marked by $M1$.
- In $\mathbf{P}\bigcirc el$ rules the conclusion $i' : A$ is marked by M_1.
- In the rules $\mathbf{A}\mathcal{U}\, el_2$ and $\mathbf{E}\mathcal{U}\, el_2$ the expression $i^{[AB]}$ is used with the following meaning: a variable i in the

derivation can be marked with $[AB]$ if it has been introduced in the derivation as a result of the application of the rule $\mathbf{A}\mathcal{U}\, el_1$ or $\mathbf{E}\mathcal{U}\, el_1$ to $i : \mathbf{A}(A\mathcal{U}\, B)$ or $\mathbf{E}(A\mathcal{U}\, B)$.

- Applying the rules $\mathbf{A}\mathcal{U}\, in_3$ or $\mathbf{E}\mathcal{U}\, in_2$ on the step n of the proof, we discard that labelled assumption, $(i \preceq j)$ or $(j \preceq l)$, which occurs earlier in the proof and all formulae until the step n.

- If an application of a rule leads to the introduction of a world variable $j \in Lab_S^{rigid}$ in its conclusion which is a relational judgement $(i \preceq j)_\varphi$, where $\varphi \in Lab_P^{rigid}$, then we mark this variable as i_φ.

- For any rule which does not require rigid world variables in its conclusion, if an application of such a rule introduces a rigid variable, say, j, in the relational judgement $(i \preceq j)_\varphi(\ \varphi \in Lab_P^{rigid})$, and j_β, for some β, then either this variable should be fresh from the list of rigid variables or the conclusion should have a form $(i \preceq j)_\beta$.

- In any rule if $(i \preceq j)_\varphi$ in $[(i \preceq j)_\varphi]*$ is an assumption, then it must be the most recent assumption that must be discarded. Applying the rule on step n of the proof, we discard $(i \preceq j)_\varphi$ and all formulae until the step n.

- Any time when we apply a rule where rigid variables are introduced in its conclusion, we pick a new variable from a list of available rigid variables. A newly introduced rigid world variable relatively binds the other variable in the relational judgement; it is similar to PLTL - this binding relation is transitive but cannot be reflexive.

- A variable which is not indicated as rigid is universal.

In addition to these we also require the following Induction Rules:

$\mathbf{A}\Box\ Induction$

$$\frac{i\!:\!A, \quad [(i \preceq j)_\varphi]*, \qquad j\!:\!A \Rightarrow \mathbf{A}\bigcirc A}{i\!:\!\mathbf{A}\Box A}$$

where $\varphi \notin Lab_P^{rigid}$, $j \notin Lab_S^{rigid}$ and $j\!:\!A \notin M1$

$\mathbf{E}\Box\ Induction$

$$\frac{i\!:\!A, \quad [(i \preceq j)_\varphi]*, \qquad j\!:\!A \Rightarrow \mathbf{E}\bigcirc A}{i\!:\!\mathbf{E}\Box A}$$

where $\varphi \notin Lab_P^{rigid}$ and $j\!:\!A \notin M1$.

We also need the following rules.

$reflexivity$

$$\frac{}{(i \preceq i)_\chi}$$

$transitivity$

$$\frac{(i \preceq j)_\chi,\ (j \preceq k)_\varphi}{(i \preceq k)_\psi}$$

where $\psi \in Lab_P^{rigid}$ is a new label, if at least one of χ or φ are elements of Lab_P^{rigid}, and $\psi \in Lab_P^{univ}$ otherwise.

$\bigcirc\ seriality$

$$\frac{}{Next(i, i')_\chi}$$

$\bigcirc / \preceq$

$$\frac{Next(i, i')_\chi}{(i \preceq i')_\chi}$$

$\prec / \preceq$

$$\frac{(i \prec j)_\chi}{(i \preceq j)_\chi}$$

Definition 10 (CTL$_{ND}$ proof) *An ND proof of a CTL formula B is a finite sequence of CTL$_{ND}$ formulae $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ which satisfies the following conditions:*

- *every $\mathcal{A}_i$ $(1 \leq i \leq n)$ is either an assumption, in which case it should have been discarded, or the conclusion of one of the ND rules, applied to some foregoing formulae,*

- *the last formula, $\mathcal{A}_n$, is $x : B$, for some label x,*

- *no rigid variable – world or path label – occurs in the conclusion or relatively binds itself.*

When B has a CTL$_{ND}$ proof we will abbreviate it as $\vdash_{ND} B$.

Examples. As examples we will prove two theorems of CTL.

Example 1.

$$\mathbf{A}\bigcirc(p \Rightarrow q) \Rightarrow (\mathbf{A}\bigcirc p \Rightarrow \mathbf{A}\bigcirc q) \qquad (1)$$

1. $x : \mathbf{A}\bigcirc(p \Rightarrow q)$	$premise$
2. $x : \mathbf{A}\bigcirc p$	$premise$
3. $Next(x, x')_\alpha$	$\bigcirc seriality$
4. $x' : p \Rightarrow q$	$\mathbf{A}\bigcirc el,\ 1, 3,\ x' \in Lab_P^{rigid}$
5. $x' : p$	$\mathbf{A}\bigcirc el,\ 2, 3$
6. $x' : q$	$\Rightarrow el,\ 4, 5$
7. $x : \mathbf{A}\bigcirc q$	$\mathbf{A}\bigcirc in,\ 3, 6$
8. $x : \mathbf{A}\bigcirc p \Rightarrow \mathbf{A}\bigcirc q$	$\Rightarrow in,\ 7,\ [2 - 7]$
9. $x : \mathbf{A}\bigcirc(p \Rightarrow q) \Rightarrow$ $(\mathbf{A}\bigcirc p \Rightarrow \mathbf{A}\bigcirc q)$	$\Rightarrow in,\ 8,\ [1 - 8]$

Example 2.

$$\mathbf{A}\Box(p \Rightarrow (\neg q \wedge \mathbf{E}\bigcirc p)) \Rightarrow (p \Rightarrow \neg\mathbf{A}(r\,\mathcal{U}\,q)) \qquad (2)$$

1. $x : \mathbf{A}\Box(p \Rightarrow (\neg q \wedge \mathbf{E}\bigcirc p)$	*premise*
2. $x : p$	*premise*
3. $x : \mathbf{A}(r\,\mathcal{U}\,q)$	*premise*
4. $(x \preceq y)_\alpha$	1, $\mathbf{A}\Box el$
5. $y : p \Rightarrow (\neg q \wedge \mathbf{E}\bigcirc p)$	1, $\mathbf{A}\Box el$
6. $y : p$	*premise*
7. $y : \neg q \wedge \mathbf{E}\bigcirc p$	$5, 6, \ \Rightarrow el$
8. $y : \neg q$	$7, \ \wedge el$
9. $y : \mathbf{E}\bigcirc p$	$7, \ \wedge el$
10. $y : p \Rightarrow \mathbf{E}\bigcirc p$	$9, \ \Rightarrow in, [6-9]$
11. $y : \mathbf{E}\Box p$	$4, 6, 10,$ *induction*
12. $x : p \Rightarrow (\neg q \wedge \mathbf{E}\bigcirc p)$	1, $\mathbf{A}\Box el$
13. $x : \neg q \wedge \mathbf{E}\bigcirc p$	$2, 12, \ \Rightarrow el$
14. $x : \neg q$	$13, \ \wedge el$
15. $(y \preceq z)_\beta$	11, $\mathbf{E}\Box el$ $\beta \in S_P^{rigid}$
16. $z : p$	11, $\mathbf{E}\Box el$
17. $(x \preceq u)_\beta$	$3, 14$ $\mathbf{A}\mathcal{U}\,el$
18. $u : q$	$3, 14$ $\mathbf{A}\mathcal{U}\,el$, $u \in S_P^{rigid}$
19. $u : p$	11, $\mathbf{E}\Box el$
20. $u : p \Rightarrow (\neg q \wedge \mathbf{E}\bigcirc p)$	1, $\mathbf{A}\Box el$
21. $u : \neg q \wedge \mathbf{E}\bigcirc p$	$19, 20, \Rightarrow el$
22. $u : \neg q$	$21, \wedge el$
23. $x : \neg\mathbf{A}(r\,\mathcal{U}\,q)$	$\neg in, 18, 22, [3-22]$
24. $x : p \Rightarrow \neg\mathbf{A}(r\,\mathcal{U}\,q)$	$\Rightarrow in, 23, [2-23]$
25. $x : (\mathbf{A}\Box(p \Rightarrow (\neg q \wedge \mathbf{E}\bigcirc p))) \Rightarrow$ $(p \Rightarrow \neg\mathbf{A}(r\,\mathcal{U}\,q))$	$\Rightarrow in, 24, [1-24]$

4 Correctness

In this section we will present the sketch of the correctness proof of our contrusction.

4.1 Soundness

Lemma 1 *Let $\Gamma = \{C_1, C_2, \ldots, C_k\}$ be a set of CTL formulae such that $\hat{\Gamma} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$, where each $\mathcal{C}_i$ $(1 \leq i \leq k)$ is $j : C_i$, for some label j, is a set of non-discarded assumptions which are contained in the CTL$_{ND}$ proof for a CTL formula B, at some step, m. Let Λ be a set of CTL$_{ND}$ formulae in the proof at step m such that for any $\mathcal{D}$, $\mathcal{D} \in \Lambda$ if it is obtained by an application of some ND rule, and let Δ be a conclusion of a CTL$_{ND}$ rule which is applied at step $m+1$. Let $\hat{\Gamma}^\star$ consist of all assumptions from $\hat{\Gamma}$ that have not been discarded by the application of this rule, the same for a set $\Lambda^\star$. Then if $\hat{\Gamma}^\star$ is realisable in a model $\mathcal{M}$ then $\Lambda^\star \cup \Delta$ is also realisable in $\mathcal{M}$.*

PROOF: We prove this lemma by induction on the number of CTL$_{ND}$ rules applied in the proof. Thus, assuming that lemma is correct for the number, n, of the CTL$_{ND}$ rules, we must show that it is also correct for the $n+1$-th rule.

The proof is quite obvious for the rules for Booleans. We only show the most interesting case where the rule of $\neg in$ is applied.

Case $\neg in$. Let $x : A$ be an element of $\hat{\Gamma}$ which is the most recent non-discarded assumption in the proof. An application of the rule $\neg in$ at step $m+1$ gives a CTL$_{ND}$ formula $x : \neg A$ as a conclusion. This means that at some earlier steps of the proof we have $y : C$ and $y : \neg C$. Here we should consider several subcases that depend on the set to which these contradictory CTL$_{ND}$ formulae belong. We now prove the lemma for some of these cases. *Subcase 1.* Assume that both $y : C$ and $y : \neg C$ are in the set $\hat{\Gamma}^\star$ but nor $y : C$ neither $y : \neg C$ coincides with $x : A$. Then the statement that the realisability of $\hat{\Gamma}^\star$ implies the realisability of $\Lambda \cup \{x : \neg A\}$ is true simply because $\hat{\Gamma}^\star$ is not realisable. *Subcase 2.* Assume that both $y : C$ and $y : \neg C$ are in the set Λ. Then if the set $\hat{\Gamma}$ realisable, the set Λ should be realisable as well. But, as assumed, it is not. So, $\hat{\Gamma}$ also can not be realisable. Note that $\hat{\Gamma} = \hat{\Gamma}^\star \cup \{x : A\}$. It should be clear that if $\hat{\Gamma}^\star$ is realisable then also $\{x : \neg A\}$ is. If we think of the set $\hat{\Gamma}$ as an initial part of the proof, then the set $\Lambda^\star$ is empty after the deletion of the corresponding steps of proof. In this case we are done.

Cases with the rules for temporal operators that do not require restrictions on labels can be shown straightforwardly from the semantics. For the cases where such a restriction is required, for example as with the $\mathbf{E}\mathcal{U}\,el$ rules, we show that given that $\Lambda^\star \cup \{j : A\}$ is realisable provided that realisability of $\hat{\Gamma}^\star$ holds. The crucial step here is to correctly define mappings f'_S and f'_P simply accurately extending the initial mapping f_S and f_P and justify these extensions.
(END)

Theorem 1 *[CTL$_{ND}$ Soundness] Let $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ be a CTL$_{ND}$ proof of CTL formula B and let $\Gamma = \{C_1, C_2, \ldots, C_n\}$ be a set of CTL formulae such that $\hat{\Gamma} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$, where each $\mathcal{C}_i$ $(1 \leq i \leq n)$ is $j : C_i$, for some label j, is a set of discarded assumptions which occur in the proof. Then $\models_{ND} B$, i.e. B is a valid formula.*

PROOF: Consider the proof $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ for some CTL formula B. According to Definition 10, $\mathcal{A}_k$ has the form $x : B$, for some label x. In general, $x : B$ belongs to some set, Λ, of non-discarded CTL$_{ND}$ formulae in the proof. By Lemma 1 we can conclude that realisability of $\hat{\Gamma}$ implies realisability of Λ. But $\hat{\Gamma}$ is empty and, therefore, is realisable in any model and for any function f by Definition 8. So Λ is also realisable in any model and for any function f. That is, any formula that belongs to Λ is valid. In particular $x : B$ is valid. (END)

4.2 Completeness

We will prove the completeness of CTL$_{ND}$ by showing that every theorem of the following axiomatics for CTL [8] is a theorem of CTL$_{ND}$.

Axioms for CTL (schemes).

1. $All\ schemes\ for\ classical\ logic$
2. $\mathbf{E}\Diamond A \equiv \mathbf{E}(\mathbf{true}\,\mathcal{U}\,A)$
3. $\mathbf{A}\,\Box A \equiv \neg\mathbf{E}\Diamond\neg A$
4. $\mathbf{A}\Diamond A \equiv \mathbf{A}(\mathbf{true}\,\mathcal{U}\,A)$
5. $\mathbf{E}\,\Box A \equiv \neg\mathbf{A}\Diamond\neg A$
6. $\mathbf{E}\bigcirc(A \vee B) \equiv \mathbf{E}\bigcirc A \vee \mathbf{E}\bigcirc A$
7. $\mathbf{A}\bigcirc A \equiv \neg\mathbf{E}\bigcirc\neg A$
8. $\mathbf{E}(A\,\mathcal{U}\,B) \Rightarrow (B \vee (A \wedge \mathbf{E}\bigcirc\mathbf{E}(A\,\mathcal{U}\,B)))$
9. $\mathbf{A}(A\,\mathcal{U}\,B) \Rightarrow (B \vee (A \wedge \mathbf{A}\bigcirc\mathbf{A}(A\,\mathcal{U}\,B)))$
10. $\mathbf{A}\bigcirc\mathbf{true} \wedge \mathbf{E}\bigcirc\mathbf{true}$
11. $\mathbf{A}\,\Box(A \Rightarrow (\neg B \wedge \mathbf{E}\bigcirc A)) \Rightarrow (A \Rightarrow \neg\mathbf{A}(C\,\mathcal{U}\,B))$
12. $\mathbf{A}\,\Box(A \Rightarrow (\neg B \wedge \mathbf{E}\bigcirc A)) \Rightarrow (A \Rightarrow \neg\mathbf{A}\Diamond B)$
13. $\mathbf{A}\,\Box(A \Rightarrow (\neg B \wedge (C \Rightarrow \mathbf{A}\bigcirc A)) \Rightarrow$
    $(A \Rightarrow \neg\mathbf{E}(C\,\mathcal{U}\,A))$
14. $\mathbf{A}\,\Box(A \Rightarrow (\neg B \wedge \mathbf{A}\bigcirc A)) \Rightarrow (A \Rightarrow \neg\mathbf{E}\Diamond B)$
15. $\mathbf{A}\,\Box(A \Rightarrow B) \Rightarrow (\mathbf{E}\bigcirc A \Rightarrow \mathbf{E}\bigcirc B)$

where $\mathbf{true}$ as an abbreviation for $\neg(p \wedge \neg p)$.

Rules:

- If $\vdash A$ and $\vdash A \Rightarrow B$ then $\vdash B$,
- If $\vdash A$ then $\vdash \mathbf{A}\,\Box A$.

To prove the completeness of CTL$_{ND}$ we first show that every instance of the scheme of the above axiomatics is a theorem of CTL$_{ND}$, and, secondly, for either of the inference rules, we establish that given that the assumptions of the rule have a CTL$_{ND}$ proof then so does its conclusion.

Lemma 2 *Every instance of the scheme of the CTL axiomatics is a theorem of CTL$_{ND}$.*

PROOF: Proofs for instances for classical schemes can be obtained by simple modifications of the corresponding proofs in the classical ND system [4]. In the previous section we proved formula (2), an instance of axiom 11. Due to the space limit we omit proofs of other instances of the axioms.
It is easy to establish the following proposition.

Proposition 1 *Let $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ be a CTL$_{ND}$ proof of a CTL formula B. Let B' be obtained from B by substituting a subformula C of B by C'. Then $\mathcal{A}'_1, \mathcal{A}'_2, \ldots, \mathcal{A}'_n$, where any occurrence of C is substituted by C' is a CTL$_{ND}$ proof of B'.*

Hence by Proposition 1 and the proofs of the instances of PLTL axioms we obtain the proof for Lemma 2.
(END)

Lemma 3 *If A has a CTL$_{ND}$ proof then $\mathbf{A}\,\Box A$ also has a CTL$_{ND}$ proof.*

Lemma 4 *If $A \Rightarrow B$ and A have CTL$_{ND}$ proofs then B also has an CTL$_{ND}$ proof.*

Proofs for these lemmas follow by showing that the desired reconstruction of the proof is always possible. In both cases the crucial step is to show that we can rewrite the proofs such that they would have completely different sets of the world and path labels.

Now we are ready to prove the completeness of CTL$_{ND}$.

Theorem 2 *[CTL$_{ND}$ Completeness] For any CTL$_{ND}$ formula, A, if $\models_{ND} A$ then there exists a CTL$_{ND}$ proof of A.*

PROOF: Consider an arbitrarily chosen theorem, A, of CTL. By induction on n, the length of the axiomatic proof for A, we now show that A also has a CTL$_{ND}$ proof.

Base Case. $n = 1$. In this case A is one of the schemes of the CTL axiomatics, and thus, the base case follows from Lemma 2.

Induction step. If Theorem 2 is correct for the proof of the length m, $(1 \leq m \leq n)$ then it is correct for the proof of the length $m + 1$.

Here the formula at the step $m + 1$ is either an axiom or is obtained from some previous formulae either by generalisation or the modus ponens rules. The proof for these cases follows from all the properties analogous to those stated in Lemma 3 and Lemma 4.

Therefore, given that A has an axiomatic proof it also has a CTL$_{ND}$ proof.
(END)

5 Discussion

We have presented a natural deduction system for the computational tree logic CTL. This will open the prospect to apply our technique as an automatic reasoning tool in a deliberative decision making framework across various AI applications where the branching-time setting is required. Although a proof-searching technique for this novel construction is still an open, and far from being trivial, problem, we expect to incorporate many of the methods previously defined for classical propositional and first-order logics. The study of complexity of the method for both classical and temporal framework, in turn, is another component of future research as well as the extension of the approach to capture more expressive branching-time frameworks such as ECTL, ECTL$^+$ and CTL*.

We believe that being equipped with the goal-directed searching procedure, based on our previous developments

[5], our technique opens broad prospects for the application of the method even in wider areas of AI and computer science, most notably, in agent engineering [13]. One of the interesting ideas of such applications of natural deduction can be found, for example, in [7]. Here the authors define a framework to reason about security protocols, and showed how the classical natural deduction system can be used as an engine for constructing valid messages. Our extension of the natural deduction to branching-time setting will be useful in managing the security protocols in a more sophisticated area of a complex dynamic environment.

References

[1] Alessandro Basso, Alexander Bolotov, Artie Basukoski, Vladimir Getov, Ludovic Henrio, and Mariusz Urbanski. Specifcation and verifcation of reconfguration protocols in grid component systems. In *To be published in the Proceedings of IS-2006*, 2006.

[2] A. Bolotov, A. Basukoski, O. Grigoriev, and V. Shangin. Natural deduction calculus for linear-time temporal logic. In *To be publsihed in the Proceedings of Jelia-2006, LNAI 4160*, 2006.

[3] A. Bolotov and M. Basukoski. Clausal resolution method for extended computation tree logic ECTL. *Journal of Applied Logic, in press*, 4(2):141–167, June 2006.

[4] A. Bolotov, V. Bocharov, A. Gorchakov, V. Makarov, and V. Shangin. *Let Computer Prove It.* Logic and Computer. Nauka, Moscow, 2004. (In Russian).

[5] A. Bolotov, V. Bocharov, A. Gorchakov, and V. Shangin. Automated first order natural deduction. In *Proceedings of IICAI*, pages 1292–1311, 2005.

[6] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronisation skeletons using branching time temporal logic. In *Logic of Programs. Proceedings of Workshop*, volume 131 of Lecture Notes in Computer Science, pages 52–71. Springer-Verlag, 1981.

[7] E.M. Clarke, S. Jha, and W. Marrero. Using state space exploration and a natural deduction style message derivation engine to verify security protocols. In *Proceedings of the IFIP Working Conference on Programming Concepts and Methods (PROCOMET)*, pages 87–106, 1998.

[8] E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Volume B, Formal Models and Semantics.*, pages 996–1072. Elsevier, 1990.

[9] E. A. Emerson and A. P. Sistla. Deciding full branching time logic. In *STOC 1984, Proceedings of*, pages 14–24, 1984.

[10] V. Makarov. Automatic theorem-proving in intuitionistic propositional logic. In *Modern Logic: Theory, History and Applications. Proceedings of the 5th Russian Conference*, StPetersburg, 1998. (In Russian).

[11] W. Quine. On natural deduction. *Journal of Symbolic Logic*, 15:93–102, 1950.

[12] A. Simpson. *The Proof Theory and Semantics of Intuitionistic Modal Logic.* PhD thesis, College of Science and Engineering, School of Informatics, University of Edinburgh, 1994.

[13] M. Wooldridge. *Reasoning about Rational Agents.* MIT Press, 2000.