Fundamenta Informaticae 72 (2006) 347–361
IOS Press

Update and Abstraction in Model Checking of Knowledge and Branching Time

N.V. Shilov[*], N.O. Garanina
Institute of Informatics Systems, Russian Academy of Science
6, Lavrentiev ave., 630090, Novosibirsk, Russia
shilov@iis.nsk.su; garanina@iis.nsk.su

K.-M. Choe
Korea Advanced Institute of Science and Technology
373-1 Kusong-dong Yusong-gu Taejon 305-701, Korea
choe@cs.kaist.ac.kr

Abstract. We present (update+abstraction) algorithm for model checking a fusion of Computation Tree Logic and Propositional Logic of Knowledge in systems with the perfect recall synchronous semantics. It has been already known that the problem is decidable with a non-elementary lower bound. The decidability follows from interpretation of the problem in a so-called Chain Logic and then in the Second Order Logic of Monadic Successors. This time we give a direct algorithm for model checking and detailed time upper bound where a number of different parameters are taken into count (i.e. a number of agents, a number of states, knowledge depth, formula size). We present a toy experiment with this algorithm that encourages our hope that the algorithm can be used in practice.

1. Introduction

Combinations of traditional program logics [19, 9, 28] with logics of knowledge [10, 27] become an actual research topic due to the importance of study of interactions between knowledge and actions for reasoning about multiagent systems. A number of techniques for (semi)automatic processing of a number of combined logics have been under study. Survey of combined logics, techniques, and results is

[*]Address for correspondence: Institute of Informatics Systems, Russian Academy of Science, 6, Lavrentiev ave., 630090, Novosibirsk, Russia

out of scope of the paper, but we would like to point out some recent research of this kind [16, 7, 8, 23, 15, 17, 18, 14].

In our previous paper [14] we addressed the model checking problem in perfect recall trace-based environments for pairwise fusion of the following program logics

(1) Elementary Propositional Dynamic Logic (EPDL),

(2) Computation Tree Logic extended by actions (Act-CTL),

(3) the propositional μ-Calculus (μC)

with the following epistemic logics

(a) Propositional Logic of Knowledge for n agents (PLK$_n$),

(b) Propositional Logic of Common Knowledge for n agents (PLC$_n$).

'Trace-based' means that semantics of formulas is defined on traces, i.e. finite sequences of states and actions. 'Synchronous' means that agents distinguish traces of different lengths. 'Perfect recall' means that every agent can distinguish different sequences of information available to him/her. If $\mathcal{L}$ stays for any of acronym of program logics EPDL, Act-CTL, or μC, and PL$\mathcal{X}$ stays for any of acronym of epistemic logics PLK$_n$ or PLC$_n$, then let acronym $\mathcal{L}$-$\mathcal{X}$ stays for fusion of logics $\mathcal{L}$ and PL$\mathcal{X}_n$. For example, Act-CTL-K$_n$ denotes fusion of Act-CTL and PLK$_n$.

It has been demonstrated in [14] that the model checking problem in the class of finitely-generated systems with perfect recall

- is $PSPACE$-complete for EPDL-C$_n$,
- is decidable for Act-CTL-K$_n$ (with a non-elementary lower bound),
- is undecidable for Act-CTL-C$_n$, μPLK$_n$ and μPLC$_n$

(where $n > 1$). These results correlate with [22] where the model checking problem for synchronous systems with perfect recall and fusion of PLK$_n$ and PLC$_n$ with Propositional Logic of Linear Time (PLLT) have been examined.

Let us observe that for the most interesting decidable case above is Act-CTL-K$_n$. The decidability in this case follows from interpretation of the problem in a so-called Chain Logic [29] and then in the Second Order Logic of Several Monadic Successors [3, 25, 24, 26, 2]. Thus it is a 'decidability in principle', a round-about in nature and is not oriented for any implementation.

In contrast we present (update+abstraction) algorithm for model checking Act-CTL-K$_n$ in perfect recall synchronous settings. We take into account a number of different parameters that contribute to algorithm complexity: a number of agents, a number of states, knowledge depth, formula size. We present a toy experiment with this algorithm that encourages our hope that the algorithm can be used in practice.

Our (update+abstraction) algorithm is inspirited by [21]. First, we define the knowledge depth for formulas of Act-CTL-K$_n$, sublogics Act-CTL-K$_n^k$ with a bounded knowledge depth $k \geq 0$, and k-trees. Intuitively, a k-tree is a finite tree of height k whose vertices are labeled by worlds of the environment and edges are labeled by agents. Observe that for every finite environment, the set $\mathcal{T}_k$ of all k-trees is finite. Then for every action a we define in an incremental manner knowledge update function G^a_k on k-trees that corresponds to change of knowledge after action. We also suggests a simple algorithm that transforms formulas of Act-CTL-K^k_n into formulas of Act^{+n}-CTL with n additional action symbols instead of agents. Finally we prove that k-trees with these update functions form a finite Kripke structure that is an abstraction of the original perfect recall environment with respect to formulas with knowledge depth k. Thus the resulting model checking algorithm simply solves formulas of Act^{+n}-CTL on k-trees.

2. Background Logics

Logics we are going to discuss are propositional polymodal logics. Semantics of these logics is defined in models which are called Kripke structures.

Definition 2.1.
Let $\{true, false\}$ be Boolean constants, *Prp* and *Rlt* be disjoint finite alphabets of propositional variables and relational symbols. Syntax of our logics consists of formulas which are constructed from Boolean constants, propositional variables, and connectives[1] $\neg$, $\wedge$, $\vee$ and some modalities.

Definition 2.2.
A model M is a triple (D_M, I_M, V_M), where the domain D_M is a nonempty set of possible worlds, the interpretation I_M maps relation symbols into binary relations on D_M, and the valuation V_M maps propositional variables into subsets of D_M. Every finite model can be considered as a labeled oriented graph with nodes and edges marked by sets of propositional variables and action symbols, respectively.

Semantics of logics is defined in terms of ternary satisfiability relation $\models$ between models, worlds and formulas.

Definition 2.3.
A satisfiability relation $\models$ between models, worlds, and formulas can be defined inductively with respect to a structure of formulas as follows. For Boolean constants $w \models_M true$ and $w \not\models_M false$ for any world w and model M. For propositional variables we have: $w \models_M p$ iff $w \in V_M(p)$. For connectives $\models$ is defined in a standard manner: $w \models_M \neg\phi$ iff $w \not\models_M \phi$, $w \models_M \phi \wedge \psi$ iff $w \models_M \phi$ and $w \models_M \psi$, $w \models_M \phi \vee \psi$ iff $w \models_M \phi$ or $w \models_M \psi$. Definition of a satisfiability relation for modalities is specific for every particular propositional polymodal logic.

A particular example of propositional polymodal logics is Propositional Logic of Knowledge (PLK) [10]. It is the simplest epistemic logic. Informally speaking, PLK is a polymodal variant of the basic propositional modal logic **S5** [4]. A special terminology, notation and models are used in this framework.

Definition 2.4. (of Propositional Logic of Knowledge for n agents PLK$_n$)
Let $n > 0$ be an integer number. An alphabet of relational symbols consists of a set of natural numbers $[1..n]$ representing names of agents. Notation for modalities is: if $i \in [1..n]$ and ϕ is a formula, then $(K_i\phi)$ and $(S_i\phi)$ are formulas[2]. For every agent $i \in [1..n]$ in every model $M = (D_M, I_M, V_M)$ interpretation $I_M(i)$ is an equivalence, i.e. a symmetric, reflexive, and transitive binary relation on D_M. Every

[1]Standard abbreviations $\rightarrow$ and $\leftrightarrow$ are admissible too.
[2]They are read as '(an agent) i knows' and '(an agent) i supposes'.

model M, where all agents in $[1..n]$ are interpreted in this way, is denoted as $(D_M, \overset{1}{\sim}, .. \overset{n}{\sim}, V_M)$ instead of (D_M, I_M, V_M) with $I_M(i) = \overset{i}{\sim}$ for every $i \in [1..n]$. In particular, for every $i \in [1..n]$ and every ϕ,

- $w \models_M (S_i\phi)$ iff for some w': $w \overset{i}{\sim} w'$ and $w' \models_M \phi$,
- $w \models_M (K_i\phi)$ iff for every w': $w \overset{i}{\sim} w'$ implies $w' \models_M \phi$.

Observe that S_i is dual[3] of K_i.

Another propositional polymodal logic Act-CTL is the basic propositional branching time temporal logic Computational Tree Logic (CTL) [9, 5, 6] extended by action symbols.

Definition 2.5. (of Act-CTL)
In the case of Act-CTL an alphabet of relational symbols consists of action symbols Act. Notation for basic modalities is: if $a \in Act$ and ϕ is a formula, then $(\mathbf{AX}^a\phi)$ and $(\mathbf{EX}^a\phi)$ are formulas. Syntax of Act-CTL has also some other special constructs associated with action symbols: if $a \in Act$, ϕ and ψ are formulas, then $(\mathbf{AG}^a\phi)$, $(\mathbf{AF}^a\phi)$, $(\mathbf{EG}^a\phi)$, $(\mathbf{EF}^a\phi)$, $(\mathbf{A}\phi\mathbf{U}^a\psi)$, and $(\mathbf{E}\phi\mathbf{U}^a\psi)$[4] are formulas too. For every $a \in Act$, an a-trace in a model M is a sequence of possible worlds $(w_1 \ldots, w_j w_{j+1} \ldots,)$ such that $(w_j, w_{j+1}) \in I_M(a)$ for every j within the sequence. (A trace can be finite.) An a-run is a maximal a-trace. (A run can be finite.) Semantics of special constructors follows[5] :

- $w \models_M \mathbf{AX}^a\phi$ iff $wrl_2 \models_M \phi$ for every a-run $wrl \in (D_M)^*$ with $wrl_1 = w$,
- $w \models_M \mathbf{EX}^a\phi$ iff $wrl_2 \models_M \phi$ for some a-run $wrl \in (D_M)^*$ with $wrl_1 = w$,
- $w \models_M \mathbf{AG}^a\phi$ iff $wrl_j \models_M \phi$ for every a-run $wrl \in (D_M)^*$ with $wrl_1 = w$
  and every $1 \le j \le |wrl|$,
- $w \models_M \mathbf{AF}^a\phi$ iff $wrl_j \models_M \phi$ for every a-run $wrl \in (D_M)^*$ with $wrl_1 = w$
  and some $1 \le j \le |wrl|$,
- $w \models_M \mathbf{EG}^a\phi$ iff $wrl_j \models_M \phi$ for some a-run with $wrl \in (D_M)^*$ $wrl_1 = w$
  and every $1 \le j \le |wrl|$,
- $w \models_M \mathbf{EF}^a\phi$ iff $wrl_j \models_M \phi$ for some a-run $wrl \in (D_M)^*$ with $wrl_1 = w$
  and some $1 \le j \le |wrl|$,
- $w \models_M \mathbf{A}(\phi\mathbf{U}^a\psi)$ iff $wrl_j \models_M \phi$ and $wrl_k \models_M \psi$
  for every a-run $wrl \in (D_M)^*$ with $wrl_1 = w$,
  for some $1 \le k \le |wrl|$ and every $1 \le j < k$,
- $w \models_M \mathbf{E}(\phi\mathbf{U}^a\psi)$ iff $wrl_j \models_M \phi$ and $wrl_k \models_M \psi$
  for some a-run $wrl \in (D_M)^*$ with $wrl_1 = w$,
  for some $1 \le k \le |wrl|$ and every $1 \le j < k$.

[3]Let us remark also that some others use another notation M_i for dual of K_i, but (in contrast to K_i), M_i has not been adopted as a standard notation.

[4]**A** is read as 'for all futures', **E** – 'for some futures', **X** – 'next time', **G** – 'always', **F** – 'sometime', **U** – 'until', and a sup-index ^{a} is read as 'in a-run(s)'.

[5]For every sequence $seq = (s_1 \ldots, s_j \ldots,)$ and every finite j within this sequence let seq_j stays for the element s_j and seq^j stays for the suffix $(s_j \ldots,)$. Operation * stays for all finite sequences.

The standard CTL is Act-CTL with a singleton alphabet Act.

3. Combining Knowledge and Branching Time

We are going to define a combined Propositional Logic of Knowledge and Branching Time Act-CTL-K$_n$.

Definition 3.1. (of Act-CTL-K$_n$)
Let $[1..n]$ be a set of agents ($n > 0$), and Act be a finite alphabet of action symbols. Syntax of Act-CTL-K$_n$ admits all knowledge modalities K_i, and S_i for $i \in [1..n]$, and all branching-time constructs **AX**^{a}, **AG**^{a}, **AF**^{a}, **AU**^{a}, **EX**^{a}, **EG**^{a}, **EF**^{a}, **EU**^{a}. Semantics is defined in terms of satisfiability $\models$. An environment is a tuple $E = (D_E, \overset{1}{\sim}, .. \overset{n}{\sim}, I_E, V_E)$ such that $(D_E, \overset{1}{\sim}, .. \overset{n}{\sim}, V_E)$ is a model for PLK$_n$ and (D_E, I_E, V_E) is a model for Act-CTL. Satisfiability is defined by induction according to semantics of propositional (Def. 2.3), knowledge (Def. 2.4), and branching time constructs (Def. 2.5). For every environment E and every formula ϕ let $E(\phi)$ be the set $\{w : w \models_E \phi\}$ of all worlds that satisfies formula ϕ in E.

We are most interested in trace-based perfect recall synchronous environments generated from background finite environments.

Definition 3.2. (of Perfect Recall Synchronous environment)
Let E be an environment $(D_E, \overset{1}{\sim}, .. \overset{n}{\sim}, I_E, V_E)$. A trace-based Perfect Recall Synchronous environment generated by E is another environment $PRS(E) = (D_{PRS(E)}, \overset{1}{\underset{\text{prs}}{\sim}}, \ldots, \overset{n}{\underset{\text{prs}}{\sim}}, I_{PRS(E)}, V_{PRS(E)})$, where

- $D_{PRS(E)}$ is the set of all pairs $(wrl, acts)$, where[6]
  $wrl \in (D_E)^+$, $acts \in Act^*$, $|wrl| = |acts| + 1$, and
  $(wrl_j, wrl_{j+1}) \in I_E(acts_j)$ for every $j \in [1..|acts|]$;
- for every $i \in [1..n]$ and for all $(wrl', acts')$, $(wrl'', acts'') \in D_{PRS(E)}$,
  $(wrl', acts') \overset{i}{\underset{\text{prs}}{\sim}} (wrl'', acts'')$ iff
  $acts' = acts''$ and $wrl'_j \overset{i}{\sim} wrl''_j$ for every $j \in [1..|wrl|]$;
- for every $a \in Act$ and for all $(wrl', acts')$, $(wrl'', acts'') \in D_{PRS(E)}$,
  $((wrl', acts'), (wrl'', acts'')) \in I_{PRS(E)}(a)$ iff[7]
  $acts'' = acts'^\wedge a$, and $wrl'' = wrl'^\wedge w''$, $(w', w'') \in I_E(a)$, where w' and w'' are the last elements in wrl' and wrl'', respectively;
- for every $p \in Prp$ and for every $(wrl, acts) \in D_{PRS(E)}$,
  $(wrl, acts) \in V_{PRS(E)}(p)$ iff $wrl_{|wrl|} \in V_E(p)$.

Importance of study of our combined logics in the framework of trace-based semantics in synchronous perfect recall settings relies upon their characteristic as logics of knowledge acquisition by actions. We would like to illustrate this characteristic and motivate our variant for synchronous perfect recall semantics by analysis of the following parameterized Guess Numbers Puzzle (GNP).

[6]Operation $^+$ stays for all non-empty finite sequences.
[7]Operation $^\wedge$ stays for the concatenation of finite words.

> Let $M, N \geq 0$ be integer parameters. Let Eloise and Abelard be two players and Orbiter be a referee. Abelard starts the game and selects a hidden number h in $[1..N]$. Abelard never reports the hidden value to Eloise. Then Eloise selects an initial value $s \in [1..N]$ for a personal counter. Eloise can increase or decrease value of the personal counter by 10, 5 or 1 while in the range $[1..N]$[8]. Eloise never reports the values of the personal counter to Abelard. Every time after operation with the counter, Orbiter reports to both players whether the new value of the personal counter is less, equal, or greater than the hidden number h. Can Eloise and Abelard simultaneously learn the hidden value h and the initial value s respectively after M increment/decrement steps?

There are two agents in the puzzle – E (Eloise) and A (Abelard): both have to learn, acquire some knowledge by experiments. These agents act in space $D = [0..N] \times [0..N] \times \{<, >, =, out, ini\} \times [1..N]$ where

- the first factor $[0..N]$ stays for an auxiliary counter c for number of increment/decrement steps,
- the second factor $[1..N]$ stays for values of the personal counter,
- the third factor $\{<, >, =, out\}$ are results of comparisons between value of the personal counter and the hidden value[9],
- the last factor $[1..N]$ stays for the hidden value.

Admissible actions for transitions from state to state are $a_{\sigma n}$, where $\sigma \in \{+, -\}$ and $n \in \{1, 5, 10\}$.

$$\underbrace{\underbrace{\underbrace{\underbrace{\boxed{0, 20, ini, 26}}_{h,\, s \in [1..100]} \xrightarrow{+10} \boxed{1, 30, >, 26}}_{h \in [20..30] \text{ and } s \in [17..100]} \xrightarrow{-5} \boxed{2, 25, <, 26}}_{h \in [25..30] \text{ and } s \in [17..30]} \xrightarrow{+1} \boxed{3, 26, =, 26}}_{\text{Knowledge of } h = 26 \text{ and } s = 20}$$

Figure 1. Example of knowledge acquisition in GNP with $N = 100$ and $h = 26$

Agent E can get knowledge about the hidden value from a sequence of states that finishes with a state with equality sign. In contrast agent A can get knowledge about the initial value from the sequence of operations that generates these sequence of states. Thus we should assume that the agents perfectly

[8]The counter does not change if its value should exit the range.
[9]out means that the action exits the legal range $[1..N]$ and hence the counter should not change.

remember both sequences. It implies that we are in synchronous perfect recall environment. Fig. 1 illustrates knowledge acquisition in a particular example of GNP (where $N = 100$ and $h = 26$). The following Act-CTL-K$_2$ formula

$$\mathbf{EF}^{next}\Big((c \leq M) \wedge$$

$$\wedge \bigvee_{h\in[1..N]} K_E(\text{hidden value is } h) \wedge \bigvee_{s\in[1..N]} K_A(\text{initial value is } s)\Big),$$

formalizes the puzzle GNP. (Here $next$ stands for the non-deterministic choice $\cup^{n\in\{1,5,10\}}_{\sigma\in\{+,-\}} a_{\sigma n}$, i.e. $next$ is an action such that $I_E(next) = \bigcup^{n\in\{1,5,10\}}_{\sigma\in\{+,-\}} I_E(a_{\sigma n}) = \{(w', w'') \mid (w', w'') \in I_E(a_{\sigma n})$ for $\sigma \in \{+, -\}$ and $n \in \{1, 5, 10\}\}$.

4. Bounded Knowledge Update

We are going to examine the model checking problem for Act-CTL-K$_n$ in perfect recall synchronous environments generated from finite environments.

Definition 4.1. (of the model checking problem)
The model checking problem for Act-CTL-K$_n$ in perfect recall synchronous environments is to decide the following set

$$CHECK(Act\text{-CTL-K}_n) \equiv \{(E,\ (wrl, acts),\ \phi) :\ E \text{ is a finite environment},\ (wrl, acts) \in D_{PRS(E)},\ \phi \text{ is a formula of } Act\text{-CTL-K}_n, \text{ and } (wrl, acts) \models_{PRS(E)} \phi\}.$$

In another words:

- the model checking problem has three inputs – a finite environment E, a trace $(wrl, acts)$, and a formula ϕ of Act-CTL-K$_n$;
- the model checking problem is to validate or refute $(wrl, acts) \models_{PRS(E)} \phi$, i.e. whether ϕ is satisfiable on $(wrl, acts)$ in E.

Let us discuss parameters that can be used for measuring of complexity. We can assume that presentation of every world has some fixed complexity, as well as presentation of every action symbol.

Definition 4.2.

- Let $E = (D_E, \overset{1}{\sim}, \ldots, \overset{n}{\sim}, I_E, V_E)$ be a finite environment represented as a finite graph: nodes are worlds D_E, edges are all pairs (w', w'') such that $w' \overset{j}{\sim} w''$ for some $j \in [1..n]$ or $(w', w'') \in I_E(a)$ for some $a \in Act$. Then let d_E be the number of worlds in D_E, and r_E be the number of edges in E; let m_E be the overall complexity $(d_E + r_E)$. For every $(wrl, acts) \in D_{PRS(E)}$ let $l_{(wrl,acts)}$ be $|wrl|$ (i.e. $|acts| + 1$).
- For every formula ϕ in Act-CTL-K$_n$ let f_ϕ be the size of ϕ (i.e. number of symbols, connectives, and constructs). A complexity measure for a tuple $(E\ ,\ (wrl, acts)\ ,\ \phi)$ is $(m_E + l_{(wrl,acts)} + f_\phi)$ (or $(m + l + f)$ whenever E, $(wrl, acts)$, and ϕ are implicit).

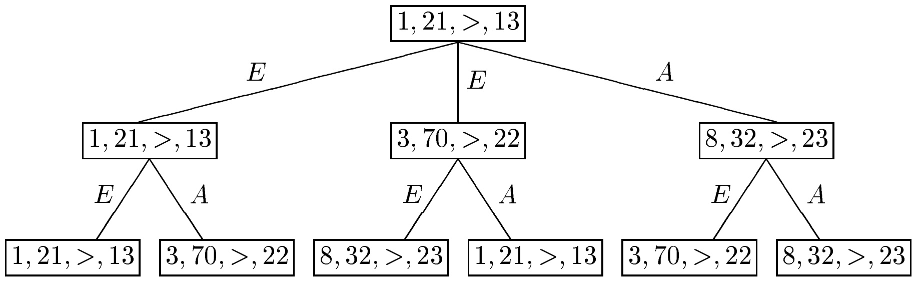

Figure 2. Example of 2-tree for GNP

The following proposition has been proved in [14]. The decidability part is due to interpretation of the model checking problem in a so-called Chain Logic [29] and then in the Second Order Logic of Several Monadic Successors [3, 25, 24, 26, 2]. The lower bound is due to interpretation of the Weak Second Order Logic of Several Monadic Successors [3, 25, 24, 26, 2] in terms of model checking.

Proposition 4.1.

For all $n > 1$ and $Act \neq \emptyset$, $CHECK(Act\text{-CTL-K}_n)$ is decidable with lower bound $\left.2^{2^{\cdot^{\cdot^{2}}}}\right\}O(t)$, where t is the overall complexity of the input.

The above Proposition 4.1 establishes non-elementary lower bound for the model checking problem for Act-CTL-K$_n$ in synchronous environments with perfect recall. It exploits the overall complexity of the problem inputs and does not distinguish a complexity impact of every input. Below we present some techniques for evaluation of a contribution of every input to this non-elementary overall complexity. First, let us define the knowledge depth for formulas of Act-CTL-K$_n$, sublogics Act-CTL-K$_n^k$ with a bounded knowledge depth $k \geq 0$, and k-trees. These definitions are inspirited by [21].

Definition 4.3.
The knowledge depth of a formula is the maximal nesting of knowledge operators in that formula. Let Act-CTL-K$_n^k$ be sublogics of Act-CTL-K$_n$ with a bounded knowledge depth $k \geq 0$.

For example, knowledge depth of $K_1(\mathbf{EX}^a K_2(q \wedge K_2 r))$ is 3. It is obvious that $Act\text{-CTL-K}_n = \bigcup_{k\geq 0} Act\text{-CTL-K}_n^k$.

For every integer $k \geq 0$ we define by mutual recursion a set $\mathcal{T}_k$ of *k-trees over E*, and a set $\mathcal{F}_k$ of *forests of k-trees over E*.

Definition 4.4.
Let $\mathcal{T}_0$ be a set of all tuples of the form $(w, \emptyset, \ldots, \emptyset)$, where w is a world and the number of copies of the empty set $\emptyset$ is equal to the number of agents n. Let us denote the world w in every tree tr by $root(tr)$. Once $\mathcal{T}_k$ has been defined, let $\mathcal{F}_k$ be the set of all subsets of $\mathcal{T}_k$. Now, define $\mathcal{T}_{k+1}$ as the set of all tuples of the form $(w, U_1, \ldots, U_n)$, where w is a world and $U_i \neq \emptyset$ is in $\mathcal{F}_k$ for each $i \in \{1..n\}$. Let us denote $\bigcup_{k\geq 0} \mathcal{T}_k$ by $\mathcal{T}$.

Intuitively, a k-tree is a finite tree of height k whose vertices are labeled by worlds of the environment E and edges are labeled by agents. In a tuple $(w, U_1, \ldots, U_n)$, the world w represents the actual state of the universe, and for each $i \in \{1..n\}$ the set U_i represents knowledge of the agent i. Identifying a 0-tree $(w, \emptyset, \ldots, \emptyset)$ with the world w, note that each component U_i in a 1-tree is simply a set of states representing knowledge of the agent i about the universe. For $k > 1$, the set U_i represents knowledge of the agent i about both the universe and knowledge of the other agents, up to the depth k. Fig. 2 represents an example of 2-tree for Guess Number Puzzle.

We would like also to remark that trees defined above are a little bit different from the trees defined in [21]. According to definition given in [21] for every $i \in \{1..n\}$ a k-tree (≥ 1) can not have any i-child, while our trees must have some. Let us remark that the main reason why we have modified the original definition is that in PLK$_n$ the nested instances of a knowledge modality K_i, $i \in \{1..n\}$, can be separated only by propositional connectives, so they are 'applied' to the same world and hence, roughly speaking, are idempotent due to an interpretation of knowledge modalities by equivalence relations. But in the case of Act-CTL-K$_n$ the nested instances of a knowledge modality can be separated by action modalities, so they may be 'applied' to different states.

Definition 4.5. Let $exp(a, b)$ be the following function:

$$exp(a,b) = \begin{cases} a, \text{ if } b = 0, \\ a \times 2^{exp(a,b-1)}, \text{ otherwise.} \end{cases}$$

Proposition 4.2.
Let $k \geq 0$ be an integer and E be a finite environment for n agents with d states. Then

- the number C_k of k-trees over E is less than or equal to $\frac{exp(n\times d,k)}{n}$.
- if $n < d$ then
  the number of nodes in every $(k+1)$-tree over E is less than $(C_k)^2$.

Proof (by induction on $k \geq 0$.)

First, $C_0 = d = exp(n \times d, 0)/n$. Let us assume that the first assertion holds for some $j \geq 0$. Then it implies the following:
$C_{j+1} \leq |\{ (w, U_1, \ldots, U_n) : w \in D,\ U_1, \ldots, U_n \in \mathcal{P}(\mathcal{T}_j) \}| = d \times (2^{C_j})^n =$
$= d \times 2^{n\times C_j} \overset{ind}{\leq} exp(n \times d, (j+1))/n.$
The proof of the first assertion is done.

Next, each 1-tree can contain at most $1 + n \times d < d^2 = (C_0)^2$ nodes. Let us assume that the second assertion holds for some $(j-1) \geq 1$. Then each $(j+1)$-tree can contain at most $1 + n \times C_j \times m_j$ nodes, where m_j is a maximal number of nodes in a j-tree. Since $n \times m_j \overset{ind}{<} d \times (C_{j-1})^2 < d \times 2^{n\times C_{j-1}} = C_j$, then the number of nodes in a $(j+1)$-tree is less than $(C_j)^2$. The proof of the second assertion is done too. □

Let $(wrl, acts)$ be a world of $PRS(E)$. Knowledge available in this world can be represented as an infinite sequence

$$tree_0(wrl, acts) \ldots, tree_k(wrl, acts) \ldots, ,$$

where each $tree_k(wrl, acts)$, $k \geq 0$, is a k-tree defined as follows.
Let $tree_0(wrl, acts)$ be $(wrl_{|wrl|}, \emptyset, \ldots, \emptyset)$, and for every $k \geq 0$ let

$tree_{k+1}(wrl,\, acts)$ be

$$\Big(wrl_{|wrl|},\ \{tree_k(wrl', acts') : (wrl', acts') \overset{1}{\underset{\text{prs}}{\sim}} (wrl, acts)\},$$
$$\ldots, \{tree_k(wrl', acts') : (wrl', acts') \overset{n}{\underset{\text{prs}}{\sim}} (wrl, acts)\} \Big).$$

Let us define some knowledge update functions for k-trees. Similar functions have been used in [21] to provide an algorithm for the model checking problem for formulas of PLK_n in synchronous environments with perfect recall.

Definition 4.6.
Let D_E and I_E be a domain and an interpretation of relation symbols from Act in the environment E. For every number $k \geq 0$, $a \in Act$ and $i \in \{1..n\}$, functions $G^a_k : \mathcal{T}_k \times D_E \to \mathcal{T}_k$ and $H^a_{k,i} : \mathcal{F}_k \times D_E \to \mathcal{F}_k$, are defined by induction on k and mutual recursion. Let

$$G^a_0(tr, w) \;=\; (w, \emptyset, \ldots, \emptyset) \text{ iff } (root(tr), w) \in I_E(a).$$

Once G^a_k has been defined, we can define for each $i \in \{1..n\}$ the function $H^a_{k,i}$ by setting $H^a_{k,i}(U, w)$ to be the set of k-trees $G^a_k(tr, w')$, where $tr \in U$ and $w' \overset{i}{\sim} w$. Using the functions $H^a_{k,i}$, $i \in \{1..n\}$, we can define G^a_{k+1} by setting $G^a_{k+1}((w, U_1, \ldots, U_n),\ w')$ to be

$$(\ w'\ , H^a_{k,1}(U_1, w'),\ \ldots,\ H^a_{k,n}(U_n, w')\ ) \text{ iff } (w, w') \in I_E(act).$$

The Fig. 3 illustrates update function G^{+10}_1 in GNP with $N = 100$ and $h = 26$.

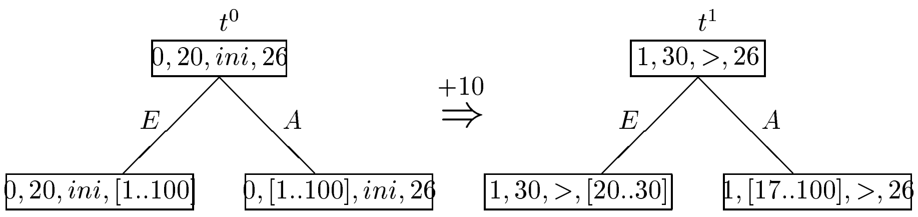

Figure 3. Example of knowledge update for $+10$ action in $TR_1(GNP)$, $N = 100$, $h = 26$

The following proposition is inspired by [21]:

Proposition 4.3.
For every $k \geq 0$, every $a \in Act$, every finite environment E, every $(wrls, acts) \in D_{PRS(E)}$, every $w \in D_E$, and every $a \in Act$, the following incremental knowledge update property holds:

$$tree_k((wrls, acts)^\wedge(w, a)) \;=\; G^a_k(tree_k(wrls, a),\ w).$$

Proof by induction on $k \geq 0$.

A basic case $k = 0$ is trivial, since 0-trees can be identified with their roots (i.e., worlds). Let us assume that the property holds for some $k > 0$. Let (ws, as), w' and a be some world of $PRS(E)$, a world of E and an action symbol in Act. Let $tree_{k+1}(ws, as) = (w, U_1, \ldots, U_n)$ and $(w, w') \in I_E(a)$. Then

$$G^a_{k+1}(tree_{k+1}(ws, as), w') =$$
$$= (w', \{G^a_k(tr, w'') : tr \in U_1, w' \overset{1}{\sim} w''\} \ldots, \{G^a_k(tr, w'') : tr \in U_n, w' \overset{n}{\sim} w''\}).$$
$$tree_{k+1}(ws^\wedge w', as^\wedge a) = (w',$$
$$\{tree_k(ws', as') : (ws', as') \underset{\text{prs}}{\overset{1}{\sim}} (ws^\wedge w', as^\wedge a)\},$$
$$\ldots,$$
$$\{tree_k(ws', as') : (ws', as') \underset{\text{prs}}{\overset{n}{\sim}} (ws^\wedge w', as^\wedge a)\}).$$

Every $G^a_k(tr, w'')$ is $tree_k(ws''{}^\wedge w'', as''{}^\wedge a)$ for some ws'', as'' in accordance with the induction assumption. But due to the definition of U_i,

$$tree_k(ws''{}^\wedge w'',\ as''{}^\wedge a) \in \{tree_k(ws', as') : (ws', as') \underset{\text{prs}}{\overset{i}{\sim}} (ws^\wedge w', as^\wedge a)\}.$$

This proves the inclusion

$$\{G^a_k(tr, w'') : tr \in U_i, w'' \overset{i}{\sim} w'\} \subseteq$$
$$\subseteq \{tree_k(ws', as') : (ws', as') \underset{\text{prs}}{\overset{i}{\sim}} (ws^\wedge w', as^\wedge a)\}.$$

The backward inclusion

$$\{G^a_k(tr, w'') : tr \in U_i, w'' \overset{i}{\sim} w'\} \supseteq$$
$$\supseteq \{\ tree_k(\ ws',\ as')\ :\ (ws',\ as') \underset{\text{prs}}{\overset{i}{\sim}} (ws^\wedge w',\ as^\wedge a)\}$$

is similar. □

5. Bounded Knowledge Abstraction

Definition 5.1.
Let Act be an alphabet of action symbols and $[1..n]$ be agents. Let Act^{+n} be $Act \cup [1..n]$, i.e., Act extended with new action symbols associated with agents $[1..n]$. A natural translation of formulas of Act-CTL-K$_n$ to formulas of Act^{+n}-CTL is simple: just replace every instance of K_i and S_i by corresponding **AX**^{i} and **EX**^{i}, respectively ($i \in [1..n]$). For every formula ϕ of Act-CTL-K$_n$, let us denote by ϕ^{+n} the resulting formula of Act^{+n}-CTL. This translation is supported by a corresponding natural transformation of environments for Act-CTL-K$_n$ to models for Act^{+n}-CTL: an environment $E = (D_E, \overset{1}{\sim}, \ldots, \overset{n}{\sim}, I_E, V_E)$ can be thought as a model $E^{+n} = (D_E, I^{+n}_E, V_E)$, where I^{+n}_E is equal to I_E on action symbols, but $I^{+n}_E(i) = \overset{i}{\sim}$ for every agent $i \in [1..n]$.

The following proposition is straightforward:

Proposition 5.1.
$E(\phi) = E^{+n}(\phi^{+n})$ for every environment E and every formula ϕ of Act-CTL-K$_n$. In particular, $PRS(E)(\phi) = (PRS(E))^{+n}(\phi^{+n})$ for every environment E and every formula ϕ of Act-CTL-K$_n$.

Observe that $PRS(E)^{+n}$ is not a single model which can be associated with the synchronous environment with perfect recall $PRS(E)$. Below we define a class of associated models based on k-trees.

Definition 5.2.
For every $k \geq 0$ let $TR_k(E)$ be a model $(D_{TR_k(E)}, I_{TR_k(E)}, V_{TR_k(E)})$ for Act^{+n}-CTL such that

- $D_{TR_k(E)}$ is the set of all 0-,..., , k-trees over E for n agents;
- $I_{TR_k(E)}(a) = \{(tr', tr'') \in D_{TR_k(E)} \times D_{TR_k(E)} :$ $tr'' = G^a_j(tr', wrl)$ for some $j \in [0..k]$ and some $wrl \in D_E$ $\}$ for $a \in Act$;
  $I_{TR_k(E)}(i) = \{(tr', tr'') \in D_{TR_k(E)} \times D_{TR_k(E)} :$ $tr'' \in U_i$ and $tr' = (wrl, U_1, \ldots, U_n)$ for some $wrl \in D_E$ $\}$ for $i \in [1..n]$;
- $V_{TR_k(E)}(p) = \{tr : root(tr) \in V_E(p)\}$ for every $p \in Prp$.

Fig. 3 and 4 illustrate interpretation of action transitions that correspond to action symbols and agents in $TR_1(GNP)$.

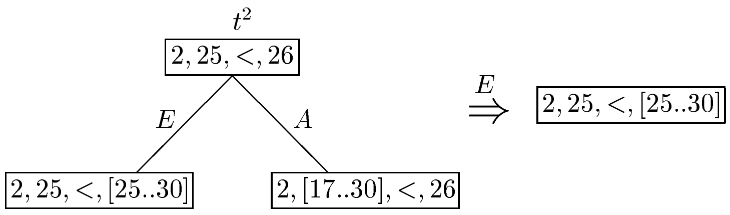

Figure 4. Example of Eloise's knowledge transition in $TR_1(GNP)$, $N = 100$, $h = 26$

Let $k \geq 0$ be an integer. Let us expand the mapping $tree_k : D_{PRS(E)} \to D_{TR_k(E)}$ on the sets of $D_{PRS(E)}$ in an element-wise manner. We would not like to distinguish between the original function $tree_k : D_{PRS(E)} \to D_{TR_k(E)}$ and its element-wise extension $tree_k : 2^{D_{PRS(E)}} \to 2^{D_{TR_k(E)}}$. A backward function $trace : 2^{D_{TR_k(E)}} \to 2^{D_{PRS(E)}}$ is also quite natural: it maps every set TR of k-trees to the set of traces $\{(wrls, acts) : tree_k(wrls, acts) \in TR\}$. The following proposition can be proved by induction on a formula structure with the help of Proposition 4.3.

Proposition 5.2.
For every $n \geq 1$ and $k \geq 0$, for every formula ϕ of Act-CTL-K$_n$ with the knowledge depth k at most, and for every finite environment E, the following holds:

- $tree_k(PRS(E)(\phi)) = TR_k(E)(\phi^{+n})$,
- $PRS(E)(\phi) = trace(TR_k(E)(\phi^{+n}))$.

This proposition can be reformulated as follows:

Proposition 5.3.
For every integer $k \geq 0$ and $n \geq 1$ and every environment E, the model $TR_k(E)$ is an abstraction of the model $(PRS(E))^{+n}$ with respect to formulas of Act^{+n}-CTL which correspond to formulas of Act-CTL-K$_n$ with the knowledge depth k at most. The corresponding abstraction function maps every trace to the k-tree of this trace.

It is well-known that the complexity of model checking of CTL and Act-CTL formulas in finite models is $O(m \times f)$, where m is an overall model complexity and f is a formula complexity. But the cited bound assumes that all worlds of the model have some constant complexity. This assumption does not hold for models where worlds are k-trees, since (in accordance with Proposition 4.2) the complexity of these trees is a non-elementary function of k and n. We should add the corresponding world-complexity factor to the complexity bound of the model checking problem. In this condition the above Propositions 4.2 and 5.3 lead to the following proposition.

Proposition 5.4.
For every integer $k \geq 1$ and $n \geq 1$, synchronous environment with perfect recall $PRS(E)$, every formula ϕ of Act-CTL-K$_n$ with the knowledge depth k at most, the model checking problem is decidable with the upper bound

$$O\Big(f \times \frac{exp(n \times d, k) \times (exp(n \times d, k-1))^2}{n^3}\Big),$$

where f is the size of the formula, d is the number of states in D_E, and the function $exp(a, b)$ is given in Definition 4.5.

The following (update+abstractin) model checking algorithm is based on the Proposition 5.3. It exploits a model checking algorithm (with complexity $O(m \times f)$) for CTL in finite models[10].

1. Let k be knowledge depth of the input formula ϕ;
2. convert ϕ into the corresponding formula ψ of Act^{+n}-CTL;
3. input a finite environment E and construct the finite model $TR_k(E)$;
4. input a trace $(wrl, acts)$ and construct the corresponding k-tree tr;
5. model check ψ on tr in $TR_k(E)$.

6. Conclusion

The above (update+abstraction)-algorithm is under process of implementation. At present a prototype[11] is in the process of testing.

Unfortunately, the known upper bound for size of $TR_k(E)$ is non-elementary function of of number of states: as we have demonstrated in our paper, an upper bound is $\frac{exp(n \times d, k)}{n}$, where d and exp are as in the above. It is really huge[12]! It implies that a straightforward use of a model checker for CTL for model checking Act^{+n}-CTL on k-trees is likely to be a non-feasible task. Roughly speaking, space $TR_k(E)$ is too big to be treated as finite.

But now there are many techniques which have been developed for infinite-state model checking. Hence, it makes sense to try to apply some of these for model checking problem Act^{+n}-CTL on k-trees.

[10]In principle any algorithm of this kind fits since we do not discuss implementation issues in this paper

[11]We call Examiner, since it checks knowledge of agents.

[12]For example, for Guess Number Puzzle the number of states $d = |D| \approx 5 \times N^3$; it implies that for formulas with knowledge depth $k = 1$ the known upper bound for size of $\mathcal{T}_k$ is $d \times 2^d \approx 5 \times N^3 \times 2^{10 \times N^3} \approx 5 \times N^3 \times 10^{3 \times N^3}$.

A very popular approach to infinite-state model checking is formalism of well-structured labeled transition systems [1, 11, 20]. Roughly speaking, a well-structured single action labeled transition system is a labeled (infinite maybe) graph provided with (pre-)order that is 'compatible' with graph edges and node labels.

It turns out that space $TR_k(E)$ provided with sub-tree partial order forms a well-structured labeled transition system where every property expressible in the propositional μ-Calculus, can be characterized by a finite computable set of maximal trees that enjoy the property. We tried feasibility of this approach to model checking of Act-CTL-K$_n$ in trace-based synchronous perfect recall synchronous environment by automatic model checking Guess Numbers Puzzle for various N, S, M (the maximal tried $N = 15$). Data structures that are used in the prototype are so-called vector-affine trees [13]. A presentation of a background theory and of our experimental model checker is a topic for a future publication.

To the best of our knowledge, the only reported (experimental) model checker for perfect recall synchronous systems is MCK [12]. It works in a linear as well as branching time settings. For perfect recall synchronous systems in temporal dimension MCK supports 'next' operator only, but neither 'always', 'sometimes', nor 'until' (although the model checking theory for the full combination of knowledge with Propositional Logic of Linear Time has been already developed [22]). The present paper has developed theory of model checking for the full combination of knowledge with branching time logics (a là Computation Tree Logic) with 'next' operator as well as with 'always', 'sometimes', and 'until'.

Acknowledgement: We would like to thanks anonymous reviewers for comments and suggestions.

References

[1] Abdulla P.A., Ĉerâns K., Jonsson B., and Tsay Yih-Kuen. Algorithmic analysis of programs with well quasi-ordered domains. Information and Computation, v.160(1-2), 2000, p.109-127.

[2] Börger E., Grädel E., Gurevich Yu. The Classical Decision Problem. Springer Verlag, 1997.

[3] Büchi J.R. On a decision method in restricted second order arithmetic. Proc. Intern. Congr. on Logic, Methodology and Philosophy of Science. Stanford Univ. Press, Stanford, 1960, p. 1–11.

[4] Bull R.A., Segerberg K. Basic Modal Logic. In: Handbook of Philosophical Logic. Vol.II. Reidel Publishing Company, 1984 (1-st ed.), Kluwer Academic Publishers, 1994 (2-nd ed.). p. 1–88.

[5] Burch J.R., Clarke E.M., McMillan K.L., Dill D.L., Hwang L.J. Symbolic Model Checking: 10^{20} states and beyond. Information and Computation, 1992. v.98(2), p. 142–170.

[6] Clarke E.M., Grumberg O., Peled D. Model Checking. MIT Press, 1999.

[7] Dixon C., Fernandez Gago M-C., Fisher M., and van der Hoek W. Using Temporal Logics of Knowledge in the Formal Verification of Security Protocols. In: Proceedings of TIME 2004, 1st-3rd July 2004, Tatihou, Normandie, France. IEEE, p.148-151.

[8] Dixon C., Nalon C. and Fisher M. Tableau for Logics of Time and Knowledge with Interactions Relating to Synchrony, Journal of Applied Non-Classical Logics, v.14, n.4, p.397-445, 2004.

[9] Emerson E.A. Temporal and Modal Logic. In: Handbook of Theoretical Computer Science. v.B, Elsevier and MIT Press, 1990, p. 995–1072.

[10] Fagin R., Halpern J.Y., Moses Y., Vardi M.Y. Reasoning about Knowledge. MIT Press, 1995.

[11] Finkel A., Schnoebelen Ph. *Well-structured transition systems everywhere!* Theor. Comp. Sci., v.256(1-2), 2001, p.63-92.

[12] Gammie P. and van der Meyden R. MCK: Model Checking the Logic of Knowledge. Springer-Verlag Lect. Notes Comp. Sci., v.3114, 2004, p.479-483.

[13] Garanina N.O. Verification of Distributed Systems on base of Affine Data representation and Logics of Knowledge and Actions. Ph.D Thesis, A.P. Ershov Institute of Informatics Systems, 2004 (in Russian).

[14] Garanina N.O., Kalinina N.A. and Shilov N.V. Model checking knowledge, actions and fixpoints. Proc. of Concurrency, Specification and Programming Workshop CS&P'2004, Germany, 2004, Humboldt Universitat, Berlin, Informatik-Bericht Nr.170, v.2, p.351-357.

[15] Halpern J. Y., van der Meyden R., and Vardi M. Y. Complete Axiomatizations for Reasoning about Knowledge and Time. SIAM Journal on Computing, v.33(3), 2004, p. 674-703.

[16] van der Hoek W. and Wooldridge M.J. Model Checking Knowledge and Time. Lecture Notes in Computer Science, v.2318, p.95-111, 2002.

[17] Kacprzak M., Lomuscio A., Penczek W. Unbounded Model Checking for Knowledge and Time. Proceedings of the CS&P'2003 Workshop, Warsaw University, v.1, p.251-264.

[18] Kacprzak M., Penczek W. Model Checking for Alternating-Time mu-Calculus via Translation to SAT. Proc. of Concurrency, Specification and Programming Workshop CS&P'2004, Germany, 2004, Humboldt Universitat, Berlin, Informatik-Bericht Nr.170, v.2, p. 286-297.

[19] Kozen D., Tiuryn J. Logics of Programs. In: Handbook of Theoretical Computer Science, v.B., Elsevier and MIT Press, 1990, p. 789–840.

[20] Kouzmin E.V., Shilov N.V., Sokolov V.A. Model Checking μ-Calculau in Well-Structured Transition Systems. Proceedings of 11th International Symposium on Temporal Representation and Reasoning (TIME 2004), France 2004, IEEE Press, p. 152-155.

[21] van der Meyden R. Common Knowledge and Update in Finite Environments. Information and Computation, 1998, v.140(2), p. 115–157.

[22] van der Meyden R., Shilov N.V. Model Checking Knowledge and Time in Systems with Perfect Recall. Springer-Verlag Lect. Notes Comput. Sci., 1999, v.1738, p. 432–445.

[23] van der Meyden R. and Wong K. Complete Axiomatizations for Reasoning about Knowledge and Branching Time. Studia Logica, v.75(1), 2003, p. 93-123.

[24] Meyer A.R. The inherent complexity of theories of ordered sets. Proc. of the Intern. Congress of Math. Vol.2. — Canadian Mathematical Congress, Vancouver, 1974, p. 477–482.

[25] Rabin M.O. Decidability of second order theories and automata on infinite trees. Trans. Amer. Math. Soc., 1969, v.141, p. 1–35.

[26] Rabin M.O. Decidable Theories. In: Handbook of Mathematical Logic. North-Holland Pub. Co., 1977, p. 595–630.

[27] Rescher N. Epistemic Logic. A Survey of the Logic of Knowledge. University of Pitsburgh Press, 2005.

[28] Shilov N.V. and Yi K. How to find a coin: propositional program logics made easy. In: Current Trends in Theoretical Computer Science, World Scientific, v. 2, 2004, p.181-213.

[29] Thomas W. Infinite trees and automaton-definable relations over ω-words. Theor. Comput. Sci., 1992, v.103, p. 143–159.