Siberian Division of the Russian Academy of Sciences
A. P. Ershov Institute of Informatics Systems

N. V. Shilov, N. O. Garanina

COMBINING KNOWLEDGE AND FIXPOINTS

Preprint
98

Novosibirsk 2002

Program logics are modal logics for reasoning about programs and systems. In general case of semantics we examine the expressive power, model checking and decidability for pairwise fusions of the following program logics: (1)Elementary Propositional Dynamic Logic, Computation Tree Logic extended by actions, and the propositional μ-Calculus with (2)Propositional Logic of Knowledge and Propositional Logic of Common Knowledge. The focus of the paper is on the study of the model checking problem for these combined logics with respect to Forgetful Asynchronous semantics and Synchronous Perfect Recall semantics.

Российская академия наук
Сибирское отделение
Институт систем информатики
им. А. П. Ершова

Н. В. Шилов, Н. О. Гаранина

КОМБИНИРОВАНИЕ ЗНАНИЙ И НЕПОДВИЖНЫХ ТОЧЕК

Препринт
98

Новосибирск 2002

В общем случае семантики исследованы выразительная сила, сложность проверки на модели и разрешимость для следующих программных логик: (1) Элементарной Пропозициональной Динамической Логики (EPDL), Ветвящейся Логики, расширенной действиями (*Act*-CTL), и Пропозиционального μ-исчисления (μC), обогащенных средствами (2) Пропозициональной Логики Знаний или Пропозициональной Логики Общих Знаний. Основным в статье является изучение задачи проверки на модели для вышеперечисленных комбинированных логик в забывающей асинхронной семантике и синхронной семантике с абсолютной памятью.

1. INTRODUCTION AND MOTIVATION

Program logics are modal logics for reasoning about programs and systems. Traditionally they comprise dynamic logics, temporal logics and their variants with explicit fixpoints [21, 10]. A recent addition to the family of program logics is a logic of knowledge [12]. In this paper we study some algorithmic properties of fusions of the propositional logic of knowledge with the propositional dynamic and state-based temporal logics extended by fixpoints. In general case of semantics we examine the expressive power (E), model checking (M) and decidability (D) for fusions of

(1) Elementary Propositional Dynamic Logic (EPDL),
(2) Computation Tree Logic extended by actions (*Act*-CTL),
(3) the propositional μ-Calculus (μC)

with

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

Results for this case can be presented in Fig. 1. In particular, we examine the model checking problem for combined logics and trace-based semantics. We are especially interested in two extreme subcases: Forgetful Asynchronous finite systems (FAS) vs. Synchronous finite systems with Perfect Recall (PRS). "Trace-based" means that possible worlds incorporate traces. "Perfect recall" means that every possible world incorporates information how it was reached, while "forgetful" means that information of this kind is not available in any world. "Synchronous" means that traces of different lengths can be distinguished, while "asynchronous" means that some traces of different lengths can be indistinguishable.

It is quite natural that in the FAS subcase neither of the discussed combined logics can express more than it can express in a background finite system. In other words, for formulae of these logics every system is

$$
\begin{array}{ccccc}
\multicolumn{5}{c}{\overbrace{\hspace{22em}}^{\text{EXPTIME-complete D}}}\\
\overset{\text{linear M}}{\text{EPDL-C}_n} & \overset{\text{E}}{<} & \overset{\text{linear M}}{\textit{Act}\text{-CTL-C}_n} & \overset{\text{E}}{<} & \overset{\text{exp. M}}{\mu\text{PLC}_n}\\
\overset{n=1}{\|}\quad\overset{n>1}{\vee} & & \overset{n=1}{\|}\quad\overset{n>1}{\vee} & & \|\\
\underbrace{\overset{\text{linear M}}{\text{EPDL-K}_n}}_{\substack{\text{PSPACE-}\\ \text{complete D}}} & \overset{\text{E}}{<} & \multicolumn{3}{c}{\underbrace{\overset{\text{linear M}}{\textit{Act}\text{-CTL-K}_n} \overset{\text{E}}{<} \overset{\text{exp. M}}{\mu\text{PLK}_n}}_{\text{EXPTIME-complete D}}}
\end{array}
$$

Fig. 1. Logic fusion summary

an abstraction of forgetful asynchronous traces generated by this system. It implies that all results represented in Fig. 1 remain valid for the FAS subcase.

In contrast, in the PRS subcase the model checking problem becomes much more complicated than in background finite systems. We show that for $n > 1$ this particular model checking problem

- is $PSPACE$-complete for EPDL-C_n,
- has non-elementary upper and lower bounds for *Act*-CTL-K_n,
- is undecidable for *Act*-CTL-C_n, μPLK$_n$ and μPLC$_n$.

These results correlate with [23] where model checking problem for synchronous systems with perfect recall and fusions of PLK$_n$ and PLC$_n$ with Propositional Logic of Linear Time (PLLT) have been examined. The cited paper has demonstrated that model checking for synchronous systems with perfect recall in the two agents case ($n = 2$)

- is $PSPACE$-complete for UNTIL-free PLLT-C_n,
- has non-elementary upper and lower bounds for PLLT-K_n,
- is undecidable for PLLT-C_n.

We should remark that our results and the results presented in [23] are closely related to [22], where $PSPACE$-completeness and undecidability have been proved for model checking formulae of PLC$_n$ in synchronous and asynchronous systems with perfect recall. The paper [23] and the present paper extend time-free results from [22] for linear/branching time.

Importance of study of combined logics in the framework of trace-based semantics in synchronous perfect recall settings rely upon their characteristic as logics of knowledge acquisition. We would like to argue this characteristic and motivate our variant for synchronous perfect recall semantics by analysis of a puzzle below. A knowledge-based analysis of a muddy children puzzle, synchronous attack and Byzantine agreement is very popular in the literature on logic of knowledge (e.g., [12]), but we would like to exploit another well-known (but hard for newcomers) puzzle:

> A set of coins consists of 14 valid and 1 false coin. All valid coins are of the same weight while the false coin has another weight. One of the valid coins is marked while others (including the false one) are unmarked. Is it possible to identify a false coin by balancing them 3 times at most?

We would like to refer to this puzzle as False Coin Puzzle. Let us generalize it as a parameterized problem $FCP(N, M)$ for $N \geq 1$ and $M \geq 0$ in such a way that the original puzzle becomes a particular instance of $FCP(N,M)$ with $N = 14$ and $M = 3$:

> A set of coins consists of $(N+1)$ enumerated coins (i.e., coins are marked by their numbers). The last coin is valid. A single coin with a number in $[1..N]$ is false, other coins with numbers in $[1..N]$ are valid. All valid coins have the same weight while weight of the false coin is different. Is it possible to identify a false coin by balancing them M times at most?

In informal analysis of $FCP(N,M)$ we would like to refer to a person who has to solve the problem as an agent. The agent does not know neither a number of a false coin, nor whether it is lighter or heavier than valid coins. Nevertheless, this number is in $[1..N]$, and the false coin is either lighter (l) or heavier (h). The agent can make balancing queries and read balancing results after each query. A balancing query $b_{(L,R)}$ is an action which consists in balancing the coins with numbers in $L \subseteq [1..N+1]$ on the left pan and coins with numbers in $R \subseteq [1..N+1]$ on the right pan (since the agent may take the valid coin $(N+1)$), where L and R are disjoint sets with $|L| = |R|$. There are three possible balancing results: $<$, $>$, and $=$, which means that the left pan is lighter, heavier than or equal to the right pan, respectively, in accordance with the number and weight of the false coin. Of course, there are initial states (marked by ini) which represent a situation when no query has been made.

Let us summarize. The agent acts in the space $[1..N] \times \{l,h\} \times$ $\times\{<,>,=,ini\}$ consisting of states. His/her admissible actions for moving between states are all $b_{(L,R)}$ for disjoint $L,R \subset [1..N+1]$ with $|L| = |R|$. The only information available for the agent in a state (i.e., which gives him/her an opportunity to distinguish between states) is a balancing result. The agent should acquire knowledge about the number of the false coin from a sequence which begins from the initial state and then consists of M queries and M correspondent results. A combination of Propositional Dynamic Logic and Propositional Logic of Knowledge seems to be a very natural framework for expressing this quest:

$$\underbrace{\langle \cup_{\dots} b_{(L,R)}\rangle \dots \langle \cup b_{(L,R)}\rangle}_{M\ \text{times}} \Bigg(\bigvee_{f\in[1..N]} K(\text{a false coin number is } f) \Bigg),$$

where all non-deterministic choices $\cup_{\dots}$ range over all $L,R \subset [1..N+1]$ such that $L \cap R = \emptyset$ and $|L| = |R|$.

$$\boxed{1,l,=} \xrightarrow{b_{\{1,2\},\{5,6\}}} \boxed{1,l,<} \qquad \boxed{1,h,<} \xrightarrow{b_{\{1,2\},\{5,6\}}} \boxed{1,h,>}$$

$$\boxed{3,l,>} \xrightarrow{b_{\{1,2\},\{5,6\}}} \boxed{3,l,=} \qquad \boxed{3,l,ini} \xrightarrow{b_{\{1,2\},\{5,6\}}} \boxed{3,l,=}$$

Fig. 2. Transitions between some states for $FCP(5,2)$

For simplicity of further discussion, let us fix some parameter's values, say, $N = 5$ and $M = 2$. In this case there are 40 states and 140 actions (some states and transitions fired by some action are depicted in Fig. 2). It is natural to assume that the agent remembers the sequence of balancing queries which he/she has done, and the sequence of corresponding balancing results, since he/she acquire knowledge about the number of the false coin from these sequences. Moreover, the agent can not distinguish between two sequences of states iff he/she has made equal sequences of queries and read equal results along these sequences. We would like to refer to this property as synchronous perfect recall hypothesis. In general, a synchronous perfect recall hypothesis is a property of an agent to remember a sequence of executed actions with a sequence of corresponding distinguishing information for intermediate states, and his/her ability to distinguish between sequences of states in accordance with these "memories". An agent knows some fact iff this fact holds in all indistinguishable situations. Fig. 3 illustrates knowledge evolution and acquisition under assumption of synchronous perfect recall.

2. BACKGROUND LOGICS

All logics we are going to discuss are some propositional polymodal logics. 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 formulae which are constructed from boolean constants, propositional variables, and connectives[1] $\neg$, $\wedge$, $\vee$ and necessity/eventuality modalities associated with relation symbols: if $r \in Rlt$ and ϕ is a formula then $(\boxed{r}\phi)$ and $(\langle r\rangle\phi)$ are formulae[2]. We would like to define semantics of logics in models, which are called Kripke structures. 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,

[1] $\rightarrow$ and $\leftrightarrow$ are admissible too but as standard abbreviations only.

[2] They are read as "box/diamond r ϕ" or "after r always/sometimes ϕ", respectively.

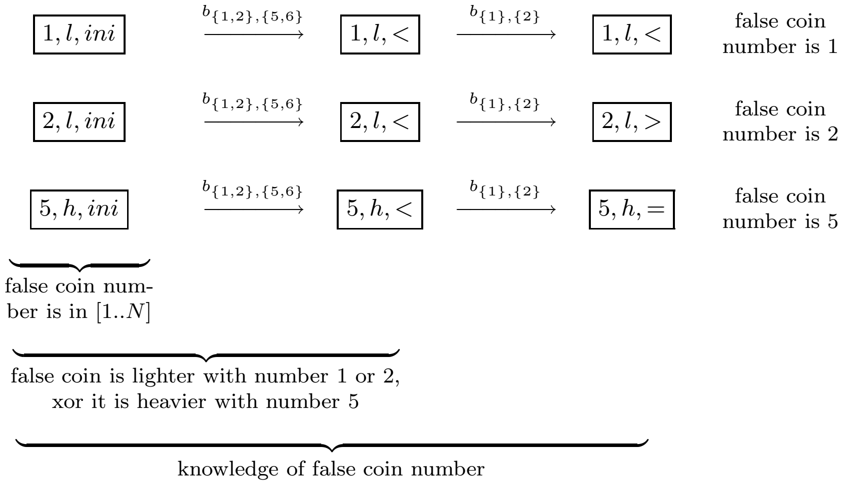

Fig. 3. Evolving knowledge in a synchronous system with perfect recall

and the valuation V_M maps propositional variables into subsets of D_M. A 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 our logics is defined in terms of satisfiability: for every model M and every formula ϕ, $M(\phi)$ is the set of all possible worlds which satisfy the formula ϕ in the model M. For every model M, every possible world w and every formula ϕ, let us write $w \models_M \phi$ iff $w \in M(\phi)$. For boolean constants $\models_M$ is defined in a standard way. For propositional variables we have: $w \models_M p$ iff $w \in V_M(p)$. For connectives $\models_M$ is defined in a standard way too. Semantics of modalities is:

- $w \models_M (\langle r \rangle \phi)$ iff $(w, w') \in I_M(r)$ and $w' \models_M \phi$ for some w',
- $w \models_M ([r] \phi)$ iff $(w, w') \in I_M(r)$ implies $w' \models_M \phi$ for every w'.

A very useful general notion is abstraction for polymodal logics. Let Φ be a set of formulae, $M_1 = (I_1, D_1)$ and $M_2 = (I_2, D_2)$ be two models and $g : D_1 \to D_2$ be a mapping. The model M_2 is called an *abstraction* [3] of the model M_1 with respect to formulae in Φ iff for any formula $\phi \in \Phi$ and any state $s \in D_1$ the following holds: $s \models_1 \phi \Leftrightarrow g(s) \models_2 \phi$.

[3] g is called an *abstraction mapping* in this case.

A particular example of propositional polymodal logics is Propositional Logic of Knowledge (PLK) [12]. A special terminology, notation and models are used in this framework. Let $n > 0$ be an integer number. An alphabet of relational symbols consists of agents $[1..n]$. Another notation for modalities is adopted: if $i \in [1..n]$ and ϕ is a formula, then $(K_i\phi)$ and $(S_i\phi)$ are formulae[4] instead of $(\boxed{i}\ \phi)$ and $(\langle i \rangle\ \phi)$. Agents should be interpreted in models by equivalence relations: $I_M(i)$ should be a symmetric, reflexive, and transitive binary relation for every $i \in [1..n]$ and every model $M = (D_M, I_M, V_M)$. Every 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_E, V_M) with $I_M(i) = \overset{i}{\sim}$ for every $i \in [1..n]$. Thus, Propositional Logic of Knowledge for n agents (PLK$_n$) is defined. We would like to note that PLK$_n$ is a polymodal variant of the basic propositional modal logic **S5** [3].

Common knowledge of several agents is a very important notion for distributed systems [12]. We would like to define Propositional Logic with Common knowledge (PLC) in the following manner. Let $n > 0$ be an integer number. In this case an alphabet of relational symbols consists of the sets of agents $2^{[1..n]}$. Another notation for modalities is adopted: if $G \subseteq [1..n]$ and ϕ is a formula, then $(C_G\phi)$ and $(J_G\phi)$ are formulae[5] instead of $\boxed{G}\phi$ and $\langle G \rangle\phi$. Agents should be interpreted in models by equivalence relations (as in PLK$_n$). Sets of agents should be interpreted in models by equivalence relations generated by participating agents: $I_M(G)$ should be a reflexive-transitive closure $\left(\bigcup_{i \in G} I_M(i)\right)^*$ for every $G \subseteq [1..n]$ and every model $M = (D_M, I_M, V_M)$. Every model M, where all agents $[1..n]$ are interpreted by equivalence relations $\overset{1}{\sim}, .. \overset{n}{\sim}$, is denoted as $(D_M, \overset{1}{\sim}, .. \overset{n}{\sim}, V_M)$ instead of (D_M, I_E, V_M) with $I_M(i) = \overset{i}{\sim}$ for every $i \in [1..n]$. We would like to adopt the same notation for PLC$_n$ models too, instead of (D_M, I_E, V_M) with $I_M(G) = \left(\bigcup_{i \in G} I_M(i)\right)^*$ for every $G \subseteq [1..n]$. Thus, Propositional Logic with Common knowledge for n agents (PLC$_n$) is defined. We would like to note that PLC$_n$ is an extension of PLK$_n$, since $C_{\{i\}}$ and $J_{\{i\}}$ are just K_i and S_i for every $i \in [1..n]$.

Elementary Propositional Dynamic Logic (EPDL) [15] is another particular propositional polymodal logic. In its framework, another special

[4] They are read as "(agent) i knows" and "(agent) i supposes".

[5] They are read as "ϕ is common knowledge of (agents) in G" and "ϕ is joint hypothesis of (agents) in G".

terminology and notation are used. In this case an alphabet of relational symbols consists of action symbols *Act*. Another notation for modalities is adopted: if $a \in Act$ and ϕ is a formula, then $([a]\phi)$ and $(\langle a\rangle\phi)$ are formulae[6] instead of $(\boxed{a}\phi)$ and $(\rlap{$\Diamond$}{\,a}\,\phi)$, respectively. But in contrast to PLK, no restriction on models is imposed. Thus, EPDL is just a polymodal variant of the basic propositional modal logic **K** [3].

Another propositional polymodal logic which we would like to define is the basic propositional branching time temporal logic Computational Tree Logic (CTL) [10, 4, 6] extended by action symbols. We would like to refer to this logic as CTL with actions (*Act*-CTL). Syntax of *Act*-CTL exploits special constructors associated with action symbols: if $a \in Act$, ϕ and ψ are formulae, then $(\mathbf{AX}^a\phi)$, $(\mathbf{EX}^a\phi)$, $(\mathbf{AG}^a\phi)$, $(\mathbf{AF}^a\phi)$, $(\mathbf{EG}^a\phi)$, $(\mathbf{EF}^a\phi)$, $(\mathbf{A}\phi\mathbf{U}^a\psi)$, and $(\mathbf{E}\phi\mathbf{U}^a\psi)$[7] are formulae too. For every sequence $seq = s_1...s_j...$ and every finite $j \leq |seq|$, let us denote an element s_j by seq_j and a suffix $s_j...$ by seq^i. For every $a \in Act$, an a-trace in a model M is a sequence of possible worlds $w_1...w_jw_{j+1}...$ such that $(w_j, w_{j+1}) \in I_M(a)$ for every component j; an a-run is a maximal a-trace. (Sic! A run can be finite.) Semantics of special constructors follows:

- $w \models_M \mathbf{AX}^a\phi$ iff $wrld_2 \models_M \phi$ for every a-run with $wrld_1 = w$,
- $w \models_M \mathbf{EX}^a\phi$ iff $wrld_2 \models_M \phi$ for some a-run with $wrld_1 = w$,
- $w \models_M \mathbf{AG}^a\phi$ iff $wrld_j \models_M \phi$ for every a-run with $wrld_1 = w$ and every $1 \leq j \leq |wrld|$,
- $w \models_M \mathbf{AF}^a\phi$ iff $wrld_j \models_M \phi$ for every a-run with $wrld_1 = w$ and some $1 \leq j \leq |wrld|$,
- $w \models_M \mathbf{EG}^a\phi$ iff $wrld_j \models_M \phi$ for some a-run with $wrld_1 = w$ and every $1 \leq j \leq |wrld|$,
- $w \models_M \mathbf{EF}^a\phi$ iff $wrld_j \models_M \phi$ for some a-run with $wrld_1 = w$ and some $1 \leq j \leq |wrld|$,
- $w \models_M \mathbf{A}(\phi\mathbf{U}^a\psi)$ iff $wrld_j \models_M \phi$ and $wrld_k \models_M \psi$ for every a-run with $wrld_1 = w$, for some $1 \leq k \leq |wrld|$ and every $1 \leq j < k$,
- $w \models_M \mathbf{E}(\phi\mathbf{U}^a\psi)$ iff $wrld_j \models_M \phi$ and $wrld_k \models_M \psi$ for some a-run with $wrld_1 = w$, for some $1 \leq k \leq |wrld|$ and every $1 \leq j < k$.

[6] They are read as "*box/diamond a* ϕ" or "*after a always/sometimes* ϕ", respectively.

[7] **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".

In particular, CTL = *Act*-CTL for a singleton alphabet *Act*. In general, *Act*-CTL is an extension of EPDL, since $\mathbf{AX}^a$ and $\mathbf{EX}^a$ are just $[a]$ and $\langle a\rangle$ for every $a \in Act$.

The propositional μ-Calculus (μC) [20, 21] is the next logic which we would like to discuss. It is EPDL extended with fixpoints. Syntax of μC expands EPDL syntax: if $p \in Prp$ and ϕ is a formula with positive instances[8] of p, then $(\mu p.\phi)$ and $(\nu p.\phi)$ are formulae[9]. Informally speaking, $\mu p.\phi$ and $\nu p.\phi$ in finite models (by the finite-case of the Tarski—Knaster fixpoint theorem) are "abbreviation" of infinite disjunctions and conjunctions:

- $false \ \vee\ \phi_p(false)\ \vee\ \phi_p(\phi_p(false))\ \vee\ \ldots\ =\ \bigvee_{i\geq 0}\phi_p^i(false)$,
- $true \ \ \wedge\ \phi_p(true)\ \ \wedge\ \phi_p(\phi_p(true))\ \ \wedge\ \ldots\ =\ \bigwedge_{i\geq 0}\phi^i(true)$,

where $\phi_p(\psi)$ is a result of substitution of a formula ψ instead of p, $\phi_p^0(\psi)$ is ψ, and $\phi_p^{i+1}(\psi)$ is $\phi_p(\phi_p^i(\psi))$ for $i \geq 0$. But in general, semantics of μ and ν can be defined in the following way. For every model M, every set $S \subseteq D_M$, and every $p \in Prp$, let us denote by $M_{S/p}$ a model which agrees with M everywhere, but $I_M(p) = S$. For every model M, every $p \in Prp$, and every formula ϕ without negative instances of p, $\lambda S.\ M_{S/p}(\phi)\ :\ S \mapsto M_{S/p}(\phi)$ is a monotonous non-decreasing mapping on 2^{D_M}. By the Tarski—Knaster theorem [27], this function has a non-empty set of fixed points $S = M_{S/p}(\phi)$ and, in particular, the least and the greatest fixed points (with respect to set inclusion $\subseteq$): $\mu(\lambda S.\ M_{S/p}(\phi))$ and $\nu(\lambda S.\ M_{S/p}(\phi))$. In these conditions

- $w \models_M (\mu p.\phi)$ iff $w \in \mu(\lambda S.\ M_{S/p}(\phi))$
  (or iff $w \in S$ for every $S \subseteq M_{S/p}(\phi)$),
- $w \models_M (\nu p.\phi)$ iff $w \in \nu(\lambda S.\ M_{S/p}(\phi))$
  (or iff $w \in S$ for some $S \supseteq M_{S/p}(\phi)$).

In general, μC is an extension of *Act*-CTL according to the following:

$$\begin{array}{ll}
\mathbf{AX}^a\phi \leftrightarrow [a]\phi & \mathbf{EX}^a\phi \leftrightarrow \langle a\rangle\phi \\
\mathbf{AG}^a\phi \leftrightarrow \nu p.\ (\phi \wedge [a]p) & \mathbf{AF}^a\phi \leftrightarrow \mu p.\ (\phi \vee [a]p) \\
\mathbf{EG}^a\phi \leftrightarrow \nu p.\ (\phi \wedge \langle a\rangle p) & \mathbf{EF}^a\phi \leftrightarrow \mu p.\ (\phi \vee \langle a\rangle p) \\
\mathbf{A}(\phi\mathbf{U}^a\psi) \leftrightarrow \mu p.\ (\psi \vee (\phi \wedge [a]p)) & \mathbf{E}(\phi\mathbf{U}^a\psi) \leftrightarrow \mu p.\ (\psi\ \vee (\phi \wedge \langle a\rangle p))
\end{array}$$

[8] I.e, p is in the range of even number of negations (otherwise it is negative instance).

[9] They are read as "mu/nu p ϕ" or "the least/greatest fixpoint p of ϕ", respectively.

3. COMBINING KNOWLEDGE AND FIXPOINTS

We are going to define a combined Propositional Logic with fixpoints and Common knowledge (μPLC). Let $[1..n]$ be a set of agents ($n > 0$), and *Act* be a finite alphabet of action symbols. Syntax of this logic admits all knowledge modalities C_G, and J_G for $G \subseteq [1..n]$, all action modalities $[a]$ and $\langle a \rangle$ for $a \in Act$, all fixpoints μp and νp for $p \in Prp$ (which are applicable to formulae with non-negative instances of p). Semantics is defined in terms of satisfiability. 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 PLC$_n$ and (D_E, I_E, V_E) is a model for μC. For every environment E and every formula ϕ, $E(\phi)$ is the set of all possible worlds which satisfy formula ϕ in E. For every environment E, every possible world w and every formula ϕ, let us write $w \models_E \phi$ iff $w \in E(\phi)$. For integrity of definition, let us briefly recall the definition of $\models$:

- $w \models_E true$ and $w \not\models_E false$;
- for $p \in Prp$, $w \models_E p$ iff $w \in V_E(p)$;
- $w \models_E (\neg\phi)$ iff $w \not\models_E \phi$;
- $w \models_E (\phi \wedge / \vee \psi)$ iff $w \models_E \phi$ and/or $w \models_E \psi$;
- for $G \subseteq [1..n]$,
  $w \models_E (C_G/J_G\phi)$ iff
  for every/some finite sequence of worlds $wrld$,
  for every/some finite sequence of agents $agn \in G^*$,
  such that
  w is the first world in $wrld$, $|wrld| = |agn| + 1$
  and $wrld_j \overset{agn_j}{\sim} wrld_{j+1}$ for all $j \in [1..|agn|]$
  implies/and $w' \models_E \phi$, where w' is the last world in $wrld$;
- for $a \in Act$ and $\{\} \equiv []/\langle\rangle$, $w \models_E (\{a\}\phi)$ iff
  $(w, w') \in I_M(a)$ implies/and $w' \models_E \phi$ for every/some w';
- $w \models_E (\mu/\nu p.\phi)$ iff $w \in \mu/\nu(\lambda S.\ E_{S/p}(\phi))$.

Thus, Propositional Logic with fixpoints and Common knowledge of n agents (μPLC$_n$) is defined. For n agents we can similarly define Propositional Logic of Knowledge with fixpoints (μPLK$_n$), as well as CTL with actions and Common knowledge (*Act*-CTL-C$_n$) and CTL with actions and Knowledge (*Act*-CTL-K$_n$), EPDL with Common knowledge (EPDL-C$_n$) and EPDL with Knowledge (EPDL-K$_n$).

Proposition 1. *All expressibilities between logics listed above are presented below:*

$n = 1$**:**

$$\begin{array}{ccccc} \textit{EPDL-C}_n & < & \textit{Act-CTL-C}_n & < & \mu\textit{PLC}_n \\ \| & & \| & & \| \\ \textit{EPDL-K}_n & < & \textit{Act-CTL-K}_n & < & \mu\textit{PLK}_n \end{array}$$

$n > 1$**:**

$$\begin{array}{ccccc} \textit{EPDL-C}_n & < & \textit{Act-CTL-C}_n & < & \mu\textit{PLC}_n \\ \vee & & \vee & & \| \\ \textit{EPDL-K}_n & < & \textit{Act-CTL-K}_n & < & \mu\textit{PLK}_n \end{array}$$

All expressibilities have linear complexity. All non-expressibilities can be justified in finite environments which represent finite games for two players.

Proof (sketch).

All the logics with knowledge can be expressed with linear complexity by their counterparts with common knowledge, since (as we have mentioned) $C_{\{i\}}$ and $J_{\{i\}}$ are just K_i and S_i for all $i \in [1..n]$. Thus, EPDL-K$_n$ $\leq$ EPDL-C$_n$, *Act*-CTL-K$_n$ $\leq$ *Act*-CTL-C$_n$, and μPLK$_n$ $\leq$ μPLC$_n$ for every $n > 0$ in general, and EPDL-K$_1$ = EPDL-C$_1$, *Act*-CTL-K$_1$ = *Act*-CTL-C$_1$, and μPLK$_1$ = μPLC$_1$ for a single agent ($n = 1$) in particular.

Next, EPDL-K$_n$ $\leq$ *Act*-CTL-K$_n$ and EPDL-C$_n$ $\leq$ *Act*-CTL-C$_n$ with linear complexity for every $n > 0$, since (as we have also mentioned) $\mathbf{AX}^a$ and $\mathbf{EX}^a$ are just $[a]$ and $\langle a \rangle$ for every $a \in \mathit{Act}$.

Then *Act*-CTL-K$_n$ $\leq$ μPLK$_n$ and *Act*-CTL-C$_n$ $\leq$ μPLC$_n$ with linear complexity for every $n > 0$, since (as we have also mentioned)

$$\begin{array}{ll} \mathbf{AX}^a\phi \leftrightarrow [a]\phi & \mathbf{EX}^a\phi \leftrightarrow \langle a \rangle\phi \\ \mathbf{AG}^a\phi \leftrightarrow \nu p.\ (\phi \wedge [a]p) & \mathbf{AF}^a\phi \leftrightarrow \mu p.\ (\phi \vee [a]p) \\ \mathbf{EG}^a\phi \leftrightarrow \nu p.\ (\phi \wedge \langle a \rangle p) & \mathbf{EF}^a\phi \leftrightarrow \mu p.\ (\phi \vee \langle a \rangle p) \\ \mathbf{A}(\phi \mathbf{U}^a \psi) \leftrightarrow \mu p.\ (\psi \vee (\phi \wedge [a]p)) & \mathbf{E}(\phi \mathbf{U}^a \psi) \leftrightarrow \mu p.\ (\psi\ \vee (\phi \wedge \langle a \rangle p)) \end{array}$$

All horizontal inequalities follow immediately from EPDL < *Act*-CTL < μC. Really, let us assume that every agent $i \in [1..n]$ has perfect knowledge about worlds, i.e. it is interpreted as identity: $\overset{i}{\sim} = =$. In this perfect knowledge case EPDL = EPDL-K$_n$ = EPDL-C$_n$, *Act*-CTL = *Act*-CTL-K$_n$ = *Act*-CTL-C$_n$, and μC = μPLK$_n$ = μPLC$_n$. For justification of inequalities in the perfect knowledge case, let P and $\neg P$ be models depicted in Fig.4. We have:

- for every EPDL formula ϕ, there exists an integer $k \geq 0$ such that $(l) \models_P \phi \Leftrightarrow (l) \models_{\neg P} \phi$ for every $l \geq k$,
  while $(m) \models_P (\mathbf{AG}^{next}p)$ and $(m) \not\models_{\neg P} (\mathbf{AG}^{next}p)$ for every $m \geq 0$;

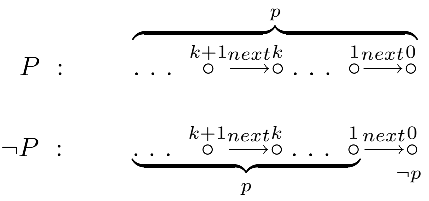

Fig. 4. Models which distinguish EPDL, *Act*-CTL and μC

thus no EPDL formula is equivalent to *Act*-CTL formula $(\mathbf{AG}^{next}p)$ (i.e., CTL formula $(\mathbf{AG}p)$);

- for every *Act*-CTL formula ϕ, there exists an integer $k \geq 0$ such that $(l) \models_P \phi \Leftrightarrow (k) \models_P \phi$ for every $l \geq k$,
  while $(2m) \models_{\neg P} (\mu q.(\langle next\rangle[next](\neg p \vee q)))$ and
  $(2m+1) \not\models_{\neg P} (\langle next\rangle[next](\neg p \vee q)))$ for every $m \geq 0$;
  thus no *Act*-CTL formula is equivalent to μC formula
  $(\mu q.(\langle next\rangle[next](\neg p \vee q)))$.

For every $k \geq 0$, finite intervals $[0..k]$ of both models P and $\neg P$ can be represented as very simple finite games P_k and $\neg P_k$ for two players, where

- positions are integers in the interval $[0..k]$,
- both players have equal possible moves $next$,
- the winning position is a position where $(\neg p)$ holds (i.e., only 0 in $\neg P$).

Thus, in a class of parameterized finite games P_k and $\neg P_k$ $(k \geq 0)$:

- EPDL can not express existence of a winning position, while *Act*-CTL can, by formula $\neg(\mathbf{AG}^{next}p)$,
- *Act*-CTL can not express existence of a winning strategy, while μC can, by formula $(\mu q.(\langle next\rangle[next](\neg p \vee q)))$.

All vertical inequalities follow immediately from $\text{PLK}_2 < \text{PLC}_2$. Really, let us assume that every action $a \in Act$ is trivial, i.e., it is interpreted as the empty set $\emptyset$. In this trivial case PLK_n = EPDL-K_n = *Act*-CTL-K_n and PLC_n = EPDL-C_n = *Act*-CTL-C_n. For justification of inequalities in the trivial case, let EO and $\neg EO$ (Even-Odd) be models depicted in Fig. 5.

- For every PLK_2-formula ϕ, there exists an integer $k \geq 0$ such that $(l) \models_{EO} \phi \Leftrightarrow (l) \models_{\neg EO} \phi$ for every $l \geq k$.
  But $(m) \models_{EO} (C_{\{1,2\}}p)$ and $(m) \not\models_{\neg EO}(C_{\{1,2\}}p)$ for every $m \geq 0$.
  Thus, no PLK_2 formula is equivalent to PLC_2 formula $(C_{\{1,2\}}p)$.

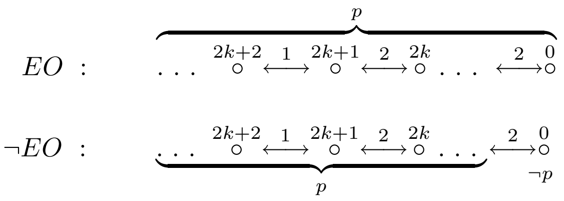

Fig. 5. Models which distinguish PLK$_n$ and PLC$_n$ for $n > 1$

For every $k \geq 0$, finite intervals $[0..k]$ of both models EO and $\neg EO$ can be presented as very simple finite games EO_k and $\neg EO_k$ for two players, where

- positions are integers in the interval $[0..k]$,
- both players have trivial moves,
- the winning position is a position where $(\neg p)$ holds (i.e., 0 in $\neg EO$).

Thus, in a class of parameterized finite games EO_k and $\neg EO_k$ $(k \geq 0)$, PLK$_2$ can not express common knowledge about existence of a winning position, while PLC$_2$ can.

Finally, μPLK$_n$ = μPLC$_n$ due to equivalencies

$$C_G\phi \leftrightarrow \nu p.(\phi \wedge (\textstyle\bigwedge_{i\in G} K_i p)) \qquad J_G\phi \leftrightarrow \mu p.(\phi \vee (\textstyle\bigvee_{i\in G} S_i p))$$

■

4. ALGORITHMIC PROBLEMS FOR COMBINED LOGICS

In general, model checking is a problem of checking whether $w \models_M \phi$ for an input world w, a model M, and formula ϕ. A particular model checking problem is to check whether $w \models_E \phi$ for an input world w, a finite environment M, and a formula ϕ of a combined logic EPDL-K$_n$, EPDL-C$_n$, ACT-CTL-K$_n$, ACT-CTL-C$_n$, μPLK$_n$, or μPLC$_n$. Let us discuss parameters used for measuring the model checking complexity in this particular case. We can assume that presentation of every world has some fixed complexity and hence we can ignore complexity of this input data. In contrast, this assumption is invalid for complexity of two other input data (a model and a formula). If E is a finite environment presented as a finite graph, then let d_E and r_E be the number of nodes and the number of edges (including knowledge); let m_E be an overall complexity $(d_E + r_E)$ of the model. If an environment E is implicit, then we would like to use these parameters without subscripts, i.e., just d, r and m. If ϕ is a formula, then let f_ϕ be a size of ϕ. Alternating fixpoint depth is another complexity parameter. If ϕ is a formula, then let a_ϕ be 1 plus the number of alternations in nesting μ and ν with respect to the syntactical dependence and positive/negative instances. Formally this parameter is defined by induction on the formula structure:

- $a_\phi = 1$ iff ϕ does not contain any fixed point,
- $a_{\phi\wedge\psi} = a_{\phi\vee\psi} = max(a_\phi, a_\psi)$,
- $a_{\langle a\rangle\phi} = a_{[a]\phi} = a_{K_i\phi} = a_{S_i\phi} = a_{C_G\phi} = a_{J_G\phi} = a_\phi$,
- $a_{\mu x.\phi} = max(a_\phi$,
  $\{(1 + a_{\nu y.\psi})$: $(\nu y.\psi)$ is a positive subformula of ϕ and ψ contains an instance of $x\}$,
  $\{(1 + a_{\mu y.\psi})$: $(\mu y.\psi)$ is a negative subformula of ϕ and ψ contains an instance of $x\})$,
- $a_{\nu x.\phi} = max(a_\phi$,
  $\{(1 + a_{\mu y.\psi})$: $(\mu y.\psi)$ is a positive subformula of ϕ and ψ contains an instance of $x\}$,
  $\{(1 + a_{\nu y.\psi})$: $(\nu y.\psi)$ is a negative subformula of ϕ and ψ contains an instance of $x\})$.

If a formula ϕ is implicit then we would like to use these parameters f_ϕ and a_ϕ without subscripts i.e., just f and a.

Proposition 2. *There exists a model checking algorithm for worlds of finite environments which runs*

- *in linear time $O(m \times f)$ for formulae of fixpoint-free logics with (common) knowledge EPDL-K_n, EPDL-C_n, Act-CTL-K_n, and Act-CTL-C_n;*
- *in exponential time $O(m \times f) \times (\frac{d\times f}{a})^{a-1}$ for formulae of logics with fixpoints and (common) knowledge μPLK_n and μPLC_n.*

Proof (sketch).

First we would like to reduce the model checking problem for all these logics in finite models to the model checking problem for μC in finite models. Then we would like to apply an efficient model checking algorithm for μC in finite models and evaluate the overall complexity for original logics. In accordance with proposition 1, all logics EPDL-K$_n$, EPDL-C$_n$, *Act*-CTL-K$_n$, *Act*-CTL-C$_n$ and μPLC$_n$ are expressible in μPLK$_n$ with linear complexity:

$$\begin{array}{ll}
\mathbf{AX}^a\phi \leftrightarrow [a]\phi & \mathbf{EX}^a\phi \leftrightarrow \langle a\rangle\phi \\
\mathbf{AG}^a\phi \leftrightarrow \nu p.\ (\phi \wedge [a]p) & \mathbf{AF}^a\phi \leftrightarrow \mu p.\ (\phi \vee [a]p) \\
\mathbf{EG}^a\phi \leftrightarrow \nu p.\ (\phi \wedge \langle a\rangle p) & \mathbf{EF}^a\phi \leftrightarrow \mu p.\ (\phi \vee \langle a\rangle p) \\
\mathbf{A}(\phi\mathbf{U}^a\psi) \leftrightarrow \mu p.\ (\psi \vee (\phi \wedge [a]p)) & \mathbf{E}(\phi\mathbf{U}^a\psi) \leftrightarrow \mu p.\ (\psi\ \vee (\phi \wedge \langle a\rangle p)) \\
C_G\phi \leftrightarrow \nu p.(\phi \wedge (\bigwedge_{i\in G} K_i p)) & J_G\phi \leftrightarrow \mu p.(\phi \vee (\bigvee_{i\in G} S_i p))
\end{array}$$

Formulae of EPDL-K$_n$ are automatically formulae of μPLK$_n$ and their alternating fixpoint depth is 1 (since they are fixpoint-free). Formulae of EPDL-C$_n$, *Act*-CTL-K$_n$, and *Act*-CTL-C$_n$ have alternating fixpoint depth 1 (since they are fixpoint-free). They are equivalent to formulae of μPLK$_n$ with alternating fixpoint depth 1, since a new bounded variable introduced in translation never occurs in subformulae.

Translation of μPLC$_n$ to μPLK$_n$ does not change the alternating fixpoint depth of formulae for the same reason as above: no bounded propositional variable introduced in translation occurs in subformulae.

Let us think of agents as auxiliary action symbols. Every environment is a model where these auxiliary action symbols are interpreted as correspondent agents. In these models formulae of μPLK$_n$ can be translated to formulae of μC with the same alternating fixpoint depth in linear time: $K_i\phi \leftrightarrow [i]\phi$ and $S_i\phi \leftrightarrow \langle i\rangle\phi$. Thus we can summarize: formulae of EPDL-K$_n$, EPDL-C$_n$, *Act*-CTL-K$_n$, *Act*-CTL-C$_n$, μPLK$_n$, and μPLC$_n$ can be translated to equivalent (in environments) formulae of μC with the same alternating fixpoint depth in linear time.

There are several correct model checking algorithms for worlds of finite models and formulae of μC. Unfortunately, $\mathcal{NP} \cap co\text{-}\mathcal{NP}$ is the best known complexity class for model checking problem for worlds of finite models and formulae of μC [11], and all known algorithms of this kind are exponential. In particular, the so-called Faster Model Checking algorithm (FMC-algorithm) has been described in [7]: FMC-algorithm is a correct model checking algorithm for μC in finite models which runs in time $O(m \times f) \times (\frac{d\times f}{a})^{a-1}$. Combining this upper bound with a linear time translation which preserves the alternating fixpoint depth, we prove the proposition. ■

Decidability is another general algorithmical problem: whether there exists a model M and a world w such that $w \models_M \phi$ for an input formula ϕ. A particular decidability problem is to check whether there exists an environment E and a world w such that $w \models_M \phi$ for an input formula ϕ

of a combined logic EPDL-K$_n$, EPDL-C$_n$, ACT-CTL-K$_n$, ACT-CTL-C$_n$, μPLK$_n$, or μPLC$_n$. For this problem the size of formulae is a single relevant complexity parameter.

Proposition 3. *EPDL-K$_n$ is $PSPACE$-complete, while EPDL-C$_n$, Act-CTL-K$_n$, Act-CTL-C$_n$, μPLK$_n$, and μPLC$_n$ are $EXPTIME$-complete.*

Proof (sketch).

It is known [12] that polymodal variants of the basic propositional modal logic **K** and **S5** are $PSPACE$-complete. EPDL-K$_n$ is $PSPACE$-hard, since it contains EPDL, i.e., polymodal **K** with a special notation. EPDL-K$_n$ is in $PSPACE$, since the decidability problem for EPDL-K$_n$ can be reduced to the decidability problem for polymodal **S5** in a linear time. We would like to reduce EPDL-K$_n$ to PLK$_{(n+2|Act|)}$ as follows.

Let $main$ and aux be new propositional variables, and for every $a \in Act$ let ind_a and err_a be new agents. For every EPDL-K$_n$ formula ϕ, let $\mathbf{S5}(\phi)$ be PLK$_{(n+2|Act|)}$ formula which results from ϕ after substitutions

- $(main \to K_{ind_a}(aux \to K_{err_a}(main \to ...)))$ instead of $([a]...)$,
- $(main \wedge S_{ind_a}(aux \wedge S_{err_a}((main \wedge ...)))$ instead of $(\langle a\rangle...)$

for every $a \in Act$. Then EPDL-K$_n$ formula ϕ is satisfiable iff PLK$_{(n+2|Act|)}$ formula $\mathbf{S5}(\phi)$ is satisfiable.

Correctness of this reduction for every EPDL-K$_n$ formula ϕ is based on the ability to simulate every binary relation by a pair of special equivalences and a pair of auxiliary monadic predicates. For simplicity of its justification, we would like to exploit a notation of Propositional Dynamic Logic (PDL) [13, 17, 21, 18] for $\mathbf{S5}(\phi)$ representation. In particular, for every PLK$_{(n+2|Act|)}$ model, $(K_i...)$ and $(S_i...)$ are equivalent to $([i]...)$ and $(\langle i\rangle...)$ for every $i \in [1..n]$, as well as for every $a \in Act$

- $(main \to K_{ind_a}(aux \to K_{err_a}(main \to ...)))$ is equivalent to $([main?;\ ind_a;\ aux?;\ err_a;\ main?]...)$,
- $(main \wedge S_{ind_a}(aux \wedge S_{err_a}((main \wedge ...)))$ is equivalent to $(\langle main?;\ ind_a;\ aux?;\ err_a;\ main?\rangle...)$.

Thus we can assume that PLK$_{(n+2|Act|)}$ formula $\mathbf{S5}(\phi)$ is written in the PDL notation. Satisfiability of PLK$_{(n+2|Act|)}$ formula $\mathbf{S5}(\phi)$ trivially implies satisfiability of EPDL-K$_n$ formula ϕ, since every $a \in Act$ can be interpreted as a PDL program $(main?;\ ind_a;\ aux?;\ err_a;\ main?)$. For validation of a backward implication, let us assume that ϕ is satisfiable in some EPDL-K$_n$ environment E. For every $a \in Act$ and all possible worlds $u, w \in D_E$, let triple (u, a, w) be a new artificial distinguishable world. Let us define a PLK$_{(n+2|Act|)}$ model M as follows:

- $D_M = D_E \cup \{(u, a, w) : u, w \in D_E,\ a \in Act\}$;
- for every $i \in [1..n]$, $\overset{i}{\sim}$ in M is $\overset{i}{\sim}$ in E extended by equality $=$ on $\{(u, a, w) : u, w \in D_E,\ a \in Act\}$;
- for every $a \in Act$, $\overset{ind_a}{\sim}$ in M is the least equivalence which contains $\{(u\ ,\ (u, a, w)) : u, w \in D_E,\ a \in Act,\ (u, w) \in I_E(a)\}$;
- for every $a \in Act$, $\overset{err_a}{\sim}$ in M is the least equivalence which contains $\{((u, a, w)\ ,\ w) : u, w \in D_E,\ a \in Act,\ (u, w) \in I_E(a)\}$;
- for every $p \in Prp$, $V_M(p) = V_E(p)$;
- $V_M(main) = D_E$ and $V_M(aux) = D_E \setminus D_M$.

In this model M, the input-output semantics of a PDL program

$$(main?;\ ind_a;\ aux?;\ err_a;\ main?)$$

is equal to $I_E(a)$ for every $a \in Act$. Hence, for every world $w \in D_E$: $w \models_E \phi$ iff $w \models_M \mathbf{S5}(\phi)$. So the PLK$_{(n+2|Act|)}$ formula $\mathbf{S5}(\phi)$ is satisfiable in the constructed model M. Thus $PSPACE$-completeness for EPDL-K$_n$ is proved.

Let us prove $EXPTIME$-completeness for *Act*-CTL-K$_n$, EPDL-C$_n$, *Act*-CTL-C$_n$, μPLK$_n$, and μPLC$_n$. $EXPTIME$-completeness is known for a variant of the basic propositional modal logic **K** extended with necessity/eventuality modalities for reflexive and transitive closure of a background single binary relation [12]. Hence *Act*-CTL-K$_n$ is $EXPTIME$-hard, since CTL can express necessity/eventuality modalities for reflexive and transitive closure by **AG** and **EF**. EPDL-C$_n$, *Act*-CTL-C$_n$, μPLK$_n$, and μPLC$_n$ are also $EXPTIME$-hard by proposition 1. The logic μPLK$_n$ is in $EXPTIME$, since the decidability problem for μPLK$_n$ can be reduced in a linear time to the decidability problem for the propositional μ-Calculus with converse (μC$^-$)[10]. In μC$^-$, if $a \in Act$ and ϕ is a formula, then $([a-]\phi)$ and $(\langle a-\rangle\phi$ are formulae such that

- $w \models_M (\langle a-\rangle\phi$ iff $(w', w) \in I_M(r)$ and $w' \models_M \phi$ for some w',
- $w \models_M ([a-]\phi)$ iff $(w', w) \in I_M(r)$ implies $w' \models_M \phi$ for every w'.

$EXPTIME$ inclusion for μC$^-$ was demonstrated in [29] three years ago. Reduction of μPLK$_n$ to μC$^-$ follows below. It implies that $EXPTIME$ includes μPLK$_n$, as well as *Act*-CTL-K$_n$, EPDL-C$_n$, *Act*-CTL-C$_n$, and

[10] This logic is a variant of μC extended with inverse of binary relations.

μPLC$_n$ (by proposition 1). For every $i \in [1..n]$ let guy_i be a new action symbol. For every μPLK$_n$ formula ϕ, let $\mathbf{MU}(\phi)$ be μC$^-$ formula which results from ϕ after substitutions

- $\nu p.(... \wedge [guy_i]p \wedge [guy_i-]p)$ instead of $(K_i...)$,
- $\mu p.(... \vee \langle guy_i\rangle p \vee \langle guy_i-\rangle p)$ instead of $(S_i...)$

for every $i \in [1..n]$. Then μPLK$_n$ formula ϕ is satisfiable iff μC$^-$ formula $\mathbf{MU}(\phi)$ is satisfiable.

Correctness of this reduction for every μPLK$_n$ formula ϕ is based on the ability to simulate every equivalence relation by a symmetric, reflexive and transitive closure of a binary relation. We would like to exploit the PDL notation for $\mathbf{MU}(\phi)$ representation. In particular, for every model and for every $i \in [1..n]$,

- $\nu p.(... \wedge [guy_i]p \wedge [guy_i-]p)$ is equivalent to $([(guy_i \cup guy_i-)*]...)$,
- $\mu p.(... \vee \langle guy_i\rangle p \vee \langle guy_i-\rangle p)$ is equivalent to $(\langle(guy_i \cup guy_i-)*\rangle...)$,

while input-output semantics of a PDL program $(guy_i \cup guy_i-)*$ is an equivalence relation. ■

Let us summarize propositions 1, 2, and 3.

Theorem 1. *The expressive power (E), model checking upper bounds (M) and decidability complexities (D) of the combined logics EPDL-K, EPDL-C, Act-CTL-K, Act-CTL-C, μPLK and μPLC in general case of semantics enjoy the properties depicted on Fig. 1.*

5. FORGETFUL ASYNCHRONOUS SYSTEMS

In this section we would like to examine trace-based forgetful asynchronous environments generated from background environments. "Trace-based" means that possible worlds are some sequences. "Forgetful" means that information about the origin and generation of worlds is not available for agents. "Asynchronous" means that traces of different lengths can be indistinguishable.

Let E be an environment $(D_E, \overset{1}{\sim}, .. \overset{n}{\sim}, I_E, V_E)$. A trace-based Forgetful Asynchronous environment generated by E is another environment $FAS(E) = (D_{FAS(E)}, \overset{1}{\underset{\text{fas}}{\approx}}, .. \overset{n}{\underset{\text{fas}}{\approx}}, I_{FAS(E)}, V_{FAS(E)})$, where

- $D_{FAS(E)}$ is D_E^+, i.e., the set of all non-empty sequences of states;

- for every $i \in [1..n]$ and for all $wrld', wrld'' \in D_{FAS(E)}$,

$$wrld' \underset{\text{fas}}{\overset{i}{\sim}} wrld'' \text{ iff } w' \overset{i}{\sim} w'',$$

  where w' and w'' are the last elements in $wrld'$ and $wrld''$, respectively;
- for every $a \in Act$ and for all $wrld', wrld'' \in D_{FAS(E)}$,

$$(wrld', wrld'') \in I_{FAS(E)}(a) \text{ iff}^{11}\ wrld'' = wrld'^{\wedge}w'', (w', w'') \in I_E(a),$$

  where w' and w'' are the last elements in $wrld'$ and $wrld''$, respectively;
- for every $p \in Prp$ and for every $wrld \in D_{FAS(E)}$,

$$wrld \in V_{FAS(E)}(p) = \text{ iff } wrld_{|wrld|} \in V_E(p).$$

Let us use some special notation in the study of the model checking problem for forgetful asynchronous systems. Let D be a set of elements. For D let

- *last-D* $: D^+ \to D$ be a function which maps every non-empty finite sequence of elements of D to the last element of this sequence;
- *D-past* $: D \to 2^{D^+}$ be a function which maps every element $d \in D$ to the set of all finite sequences with this last element d.

Both functions can be extended on power-sets in the standard manner:

- *last-D* $: 2^{D^+} \to 2^D$ be a function which maps every set of non-empty finite sequences to the set of last elements of these sequences;
- *D-past* $: 2^D \to 2^{D^+}$ be a function which maps every set of elements to the set of all finite sequences with these last elements.

We would not distinguish these functions from their extensions.

Proposition 4. *For every formula ϕ of the combined Propositional Logic with fixpoints and Knowledge μPLK and for every environment E, the following holds:*

- $FAS(E)(\phi) = D_E\text{-}past(E(\phi))$,
- $E(\phi) = last\text{-}D_E(FAS(E)(\phi))$

(i.e., the formula ϕ holds on a non-empty finite sequence of worlds in the corresponding forgetful asynchronous environment $FAS(E)$ iff ϕ holds on the last world of the sequence in the background environment E).

[11] Here $^{\wedge}$ is concatenation of words.

Proof (sketch) by induction on the formula structure.

A basic case for elementary formulae *true* and *false* is trivial:

- $$\begin{aligned} FAS(E)(true) = & \; D_{FAS(E)} = D_E^+ = D_E\text{-}past(D_E) \\ = & \; D_E\text{-}past(E(true)), \end{aligned}$$
- $E(false) = \emptyset = last\text{-}D_E(\emptyset) = last\text{-}D_E(FAS(E)(false))$

for every environment E. Another basic case for propositional variables immediately follows from the definition of a valuation for forgetful asynchronous semantics:

- $$\begin{aligned} FAS(E)(p) = & \; V_{FAS(E)}(p) = \{wrld \in D_{FAS(E)} : \; wrld_{|wrld|} \in V_E(p)\} \\ = & \; D_E\text{-}past(V_E(p)) = D_E\text{-}past(E(p)), \end{aligned}$$
- $$\begin{aligned} E(p) = & \; V_E(p) = last\text{-}D_E(\{wrld \in D_{FAS(E)} : \; wrld_{|wrld|} \in V_E(p)\}) \\ = & \; last\text{-}D_E(V_{FAS(E)}(p)) = last\text{-}D_E(FAS(E)(p)). \end{aligned}$$

for every environment E and every propositional variable p.

Inductive cases can be classified as follows:

1. $\phi \equiv (\neg\psi)$ for some ψ,
2. (a) $\phi \equiv (\psi_1 \wedge \psi_2)$ or (b) $\phi \equiv (\psi_1 \vee \psi_2)$ for some ψ_1 and ψ_2,
3. (a) $\phi \equiv ([a]\psi)$ or (b)$\phi \equiv (\langle a\rangle\psi)$ for some action symbol a and some ψ,
4. (a) $\phi \equiv (K_i\psi)$ or (b) $\phi \equiv (S_i\psi)$ for some agent i and some ψ,
5. (a) $\phi \equiv (\mu p.\psi)$ or (b) $\phi \equiv (\nu p.\psi)$ for some variable p and some ψ.

Below we sketch out one subcase of each of these cases (since proofs for both subcases are similar to each other but differ from case to case). We use notation $\overset{ind}{=}$ for some equalities when they hold due to the induction assumption.

Inductive case 1: $\phi \equiv (\neg\psi)$ for some ψ. For every environment E the following holds:

- $$\begin{aligned} FAS(E)(\phi) = & \; D_{FAS(E)} \setminus FAS(E)(\psi) \overset{ind}{=} D_E^+ \setminus D_E\text{-}past(E(\psi)) \\ = & \; D_E^+ \setminus \{wrld \in D_E^+ \; : \; wrld_{|wrld|} \in E(\psi)\} \\ = & \; \{wrld \in D_E^+ \; : \; wrld_{|wrld|} \notin E(\psi)\} \\ = & \; \{wrld \in D_E^+ \; : \; wrld_{|wrld|} \in E(\phi)\} \\ = & \; D_E\text{-}past(E(\phi)); \end{aligned}$$
- $$\begin{aligned} E(\phi) \; & = D_E \setminus E(\psi) \overset{ind}{=} D_E \setminus last\text{-}D_E(FAS(E)(\psi)) \\ & \overset{ind}{=} last\text{-}D_E(D_E^+ \setminus FAS(E)(\psi)) = last\text{-}D_E(FAS(E)(\phi)). \end{aligned}$$

Inductive case 2(a): $\phi \equiv (\psi_1 \wedge \psi_2)$. For every environment E the following holds:

- $$\begin{aligned} FAS(E)(\phi) &= FAS(E)(\psi_1) \cap FAS(E)(\psi_2) \\ &\overset{ind}{=} D_E\text{-}past(E(\psi_1)) \cap D_E\text{-}past(E(\psi_2)) \\ &= \{wrld \in D_E^+ \ : \ wrld_{|wrld|} \in E(\psi_1)\} \\ &\qquad \cap \{wrld \in D_E^+ \ : \ wrld_{|wrld|} \in E(\psi_2)\} \\ &\overset{ind}{=} \{wrld \in D_E^+ \ : \ wrld_{|wrld|} \in E(\phi)\} \\ &= D_E\text{-}past(E(\phi)); \end{aligned}$$
- $$\begin{aligned} E(\phi) &= E(\psi_1) \cap E(\psi_2) \\ &\overset{ind}{=} last\text{-}D_E(FAS(E)(\psi_1)) \cap last\text{-}D_E(FAS(E)(\psi_2)) \\ &\overset{ind}{=} last\text{-}D_E(FAS(E)(\psi_1) \cap FAS(E)(\psi_2))) \\ &= last\text{-}D_E(FAS(E)(\phi)). \end{aligned}$$

Inductive case 3(b): $\phi \equiv (\langle a \rangle \psi)$. For every environment E the following holds:

- $$\begin{aligned} FAS(E)(\phi) = &\ \{wrld^\wedge w' \ : \ wrld \in D_E^*, w' \in D_E \\ &\quad \text{and } (wrld^\wedge w' \, ^\wedge w'') \in FAS(E)(\psi) \\ &\qquad \text{for some } w'' \in D_E \text{ such that } (w', w'') \in I_E(a)\} \\ \overset{ind}{=} &\ \{wrld^\wedge w' \ : \ wrld \in D_E^*, \ w' \in D_E \text{ and } w'' \in E(\psi) \\ &\quad \text{for some } w'' \in D_E \text{ such that } (w', w'') \in I_E(a)\} \\ = &\ \{wrld^\wedge w' \ : \ wrld \in D_E^*, \ w' \in D_E \text{ and } w' \in E(\langle a \rangle \psi)\} \\ = &\ \{wrld \in D_E^+ \ : \ wrld_{|wrld|} \in E(\phi)\} = D_E\text{-}past(E(\phi)); \end{aligned}$$
- $$\begin{aligned} E(\phi) = &\ \{w' \ : \ w' \in D_E \text{ and } w'' \in E(\psi) \\ &\quad \text{for some } w'' \in D_E \text{ such that } (w', w'') \in I_E(a)\} \\ \overset{ind}{=} &\ last\text{-}D_E\big(\{wrld^\wedge w' \ : wrld \in D_E^*, \ w' \in D_E \\ &\quad \text{and } (wrld^\wedge w' \, ^\wedge w'') \in FAS(E)(\psi) \\ &\qquad \text{for some } w'' \in D_E \text{ such that } (w', w'') \in I_E(a)\}\big) \\ = &\ last\text{-}D_E(FAS(E)(\phi)). \end{aligned}$$

Inductive case 4(b): $\phi \equiv (S_i \psi)$. For every environment E the following holds:

- $$\begin{aligned} FAS(E)(\phi) = &\ \{wrld'^\wedge w' \ : wrld' \in D_E^*, w' \in D_E \\ &\quad \text{and } (wrld''^\wedge w'') \in FAS(E)(\psi) \\ &\qquad \text{for some } wrld'' \in D_E^* \\ &\qquad\quad \text{and } w'' \in D_E \text{ such that } w' \overset{i}{\sim} w''\} \\ \overset{ind}{=} &\ \{wrld' \, ^\wedge w' \ : \ wrld' \in D_E^*, \ w' \in D_E \text{ and } w'' \in E(\psi) \\ &\quad \text{for some } w'' \in D_E \text{ such that } w' \overset{i}{\sim} w''\} \\ = &\ \{wrld' \, ^\wedge w' \ : \ wrld \in D_E^*, \ w' \in D_E \text{ and } w' \in E(\phi)\} \\ = &\ \{wrld \in D_E^+ \ : \ wrld_{|wrld|} \in E(\phi)\} = D_E\text{-}past(E(\phi)); \end{aligned}$$

- $E(\phi) = \{w' \ : \ w' \in D_E \text{ and } w'' \in E(\psi)$
  $\text{for some } w'' \in D_E \text{ such that } w' \overset{i}{\sim} w''\}$
  $\overset{ind}{=} \ \textit{last-}D_E\big(\{wrld'\ {}^\wedge w' \ : wrld' \in D^*_E,\ w' \in D_E$
  $\text{and } (wrld''\ {}^\wedge w'') \in FAS(E)(\psi) \text{ for some } wrld'' \in D^*_E$
  $\text{and } w'' \in D_E \text{ such that } w' \overset{i}{\sim} w''\}\big)$
  $= \ \textit{last-}D_E(FAS(E)(\phi)).$

Inductive case 5(a): (a) $\phi \equiv (\mu p.\psi)$. In accordance with the Tarski-Knaster theorem [27], the least fixpoint $\mu(\lambda s.\ f(s))$ of a non-decreasing mapping $(\lambda s.\ f(s)) : S \to S$ over a complete lattice S is equal to the least upper bound $lub_\alpha f^\alpha(\bot)$, where $\bot$ is the least element of S, an ordinal α is less than an ordinal number for S, and for every ordinal α within this range

$$f^\alpha(\bot) \ = \ \begin{cases} \bot, \text{ if } \alpha = 0, \\ f(f^\beta(\bot)), \text{ if } \alpha = \beta + 1 \text{ for some } \beta, \\ lub_{\beta<\alpha} f^\beta(\bot), \text{ otherwise.} \end{cases}$$

Let E be an environment. In this particular case we have

$E(\phi)$:	$FAS(E)(\phi)$:
$S = 2^{D_E}$	$S = 2^{D^+_E}$
$(\lambda s.\ f(s)) \ =$ $= \ \big(\lambda W \subseteq D_E.\ (E_{W/p}(\psi)\big)$	$(\lambda s.\ f(s)) \ = \ \big(\lambda WRLD \subseteq D^+_E.$ $(FAS(E)_{WRLD/p}(\psi)\big)$

(the set-theoretic union $\bigcup$ is the least upper bound in S the empty set $\emptyset$ is the least element in S)

For every ordinal α, let

$$E^\alpha \ = \ \begin{cases} E_{\bot/p}, \text{ if } \alpha = 0, \\ E_{(E^\beta(\psi))/p}, \text{ if } \alpha = \beta + 1 \text{ for some } \beta, \\ E_{(\cup_{\beta<\alpha} E^\beta(\psi))/p}, \text{ otherwise;} \end{cases}$$

$$FAS(E)^\alpha \ = \ \begin{cases} FAS(E)_{\bot/p}, \text{ if } \alpha = 0, \\ FAS(E)_{(FAS(E)^\beta(\psi))/p}, \text{ if } \alpha = \beta + 1 \text{ for some } \beta, \\ FAS(E)_{(\cup_{\beta<\alpha} FAS(E)^\beta(\psi))/p}, \text{ otherwise.} \end{cases}$$

Due to the induction assumption for ψ, it is easy to prove by the ordinal induction that $FAS(E)^\alpha = FAS(E^\alpha)$ for every ordinal α. It implies that

$$\textit{last-}D_E(FAS(E)^\alpha(\psi)) \ = \ E^\alpha(\psi) \text{ and } D_E\textit{-past}(E^\alpha(\psi)) \ = \ FAS(E)^\alpha(\psi)$$

also for every ordinal α. It implies that

$$\textit{last-}D_E(FAS(E)(\phi)) = E(\phi) \text{ and } D_E\textit{-past}(E(\phi)) = FAS(E)(\phi),$$

since $\phi \equiv \mu p.\psi$ and

$$E(\mu p.\psi) = E^{\alpha}(\psi)\ ,\ FAS(E)(\mu p.\psi) = FAS(E)^{\alpha}(\psi)$$

for some ordinal α in accordance with the Tarski—Knaster theorem.

This finishes the proof of the inductive case (5a). But we would like to demonstrate below how to exploit the theory of abstract interpretation[12] [8, 9] instead of the direct use of the Tarski—Knaster theorem.

First let us recall some rudiments of the related theory.

Let A and B be partially ordered sets, $f : A \to B$ and $g : B \to A$ be total mappings. These mappings are a Galois connection iff for every $a \in A$ and $b \in B$:

$$f(a) \overset{B}{\preceq} b \ \Leftrightarrow\ a \overset{A}{\preceq} g(b).$$

If these mappings are a Galois connection, then the following standard notation $A \underset{f}{\overset{g}{\leftrightarrows}} B$ is used. The following least fixpoint property has been proven in [9]:

> Let A and B be complete partial orders, $A \underset{f}{\overset{g}{\leftrightarrows}} B$ be a Galois connection with continuous f such that $F(\bot_A) = \bot_B$, and $F : A \to A$ and $G : B \to B$ be monotone mappings such that $f \circ F = G \circ f$. Then $f(\mu F) = \mu G$.

Let E be an environment. The sets $C = 2^{D_E}$ and

$$D = \{WRLD \subseteq D_E^{+} \ :\ \text{for all } wrld \in WRLD \text{ and } wrld' \in D_E^{+} \text{ if } wrld_{|wrld|} = wrld'_{|wrld'|} \text{ then } wrld' \in WRLD\ \}$$

are complete partial orders. Let P be $\lambda WRLD \in D\ .\ FAS(E)_{WRLD/p}(\psi)$ and R be $\lambda W \in C\ .\ E_{W/p}(\psi)$. These P and R are monotone mappings.

- $D \underset{\textit{last-}D}{\overset{D\textit{-past}}{\leftrightarrows}} C$ is a Galois connection. Let us consider

$$D \text{ as } A \text{ and } C \text{ as } B,$$
$$P \text{ as } F \text{ and } R \text{ as } G,$$
$$\textit{last-}D_E \text{ as } f.$$

[12] Do not mix with abstraction from section 2.

This f is a continuous function which preserves the least elements in A and B. For every $S \in D$ the following holds due to the induction assumption for ψ:

$$\begin{aligned}(f \circ F)(S) = & \ \big(\mathit{last}\text{-}D_E(\lambda WRLD \in D \ . \ FAS(E)_{WRLD/p}(\psi))\big)(S) \\ \overset{ind}{=} & \ \big(\lambda W \in C \ . \ E_{(W/p}(\psi)\big)\big(\mathit{last}\text{-}D_E(S)\big) = (G \circ f)(S).\end{aligned}$$

Hence $\mathit{last}\text{-}D_E(FAS(E)(\phi)) = \mu(f \circ F) = \mu G = E(\phi)$ in accordance with the least fixpoint image property.

- $C \ \underset{D_E\text{-}past}{\overset{last\text{-}D_E}{\overset{\longleftarrow}{\longrightarrow}}} \ D$ is also a Galois connection. Let us consider

  C as A and D as B,
  R as F and P as G,
  D_E-*past* as f.

  This f is a continuous function which preserves the least elements in A and B. For every $S \in C$ the following holds due to the induction assumption for ψ:

$$\begin{aligned}(f \circ G)(S) = & \ \big(D_E\text{-}\mathit{past}(\lambda W \in C \ . \ E_{W/p}(\psi))\big)(S) \\ \overset{ind}{=} & \ \big(\lambda WRLD \in D \ . \ FAS(E)_{WRLD/p}(\psi)\big)\big(D_E\text{-}\mathit{past}(S)\big) \\ = & \ (F \circ f)(S).\end{aligned}$$

  Hence $D_E\text{-}\mathit{past}(E(\phi)) = \mu(f \circ G) = \mu F = FAS(E)(\phi)$ in accordance with the least fixpoint image property.

■

From proposition 4, we can deduce the following

Proposition 5. *Every environment E is an abstraction of the corresponding forgetful asynchronous environment $FAS(E)$ with respect to formulae of the combined Propositional Logic with fixpoints and Common knowledge μPLC. The corresponding abstraction function maps every non-empty finite sequence of states to the last element of the sequence.*

This proposition together with Theorem 1 implies the following theorem.

Theorem 2. *The expressive power, the model checking problem and decidability of the combined logics EPDL-K_n, EPDL-C_n, Act-CTL-K_n, Act-CTL-C_n, μPLK_n and μPLC_n in forgetful asynchronous settings are equivalent to the expressive power, the model checking problem and decidability of these logics in general case of semantics.*

$$\overbrace{\overset{\textit{linear }M}{\textit{EPDL-}C_n}\ \overset{E}{<}\ \overset{\textit{linear }M}{\textit{Act-CTL-}C_n}\ \overset{E}{<}\ \overset{\textit{exp. }M}{\mu PLC_n}}^{\textit{EXPTIME-complete }D}$$

$$\begin{array}{ccc} \overset{n=1}{\parallel}\quad\overset{n>1}{\vee} & \overset{n=1}{\parallel}\quad\overset{n>1}{\vee} & \parallel \end{array}$$

$$\underbrace{\overset{\textit{linear }M}{\textit{EPDL-}K_n}}_{\substack{\textit{PSPACE-}\\ \textit{complete }D}}\ \overset{E}{<}\ \underbrace{\overset{\textit{linear }M}{\textit{Act-CTL-}K_n}\ \overset{E}{<}\ \overset{\textit{exp. }M}{\mu PLK_n}}_{\textit{EXPTIME-complete }D}$$

Let us conclude this section with a remark on an admissible modification of the definition of forgetful asynchronous systems, which does not change the algorithmic properties. In the above original settings, information about origin and generation of worlds is not available for agents in forgetful semantics: an agent can not distinguish between two traces iff their last states are indistinguishable. In other words, the agents are people who do not know either their own history or that they have a history. It turns out that it is acceptable (without violating propositions 4, 5 and 2) to afford them to remember that they have some history. In this new case of semantics an agent can not distinguish between two traces iff their last states are indistinguishable (as in the original settings) and both traces are generated by some (may be different) sequences of actions.

6. SYNCHRONOUS SYSTEMS WITH PERFECT RECALL

We are especially interested in trace-based perfect recall synchronous environments generated from background finite environments. Here "trace-based" means that possible worlds incorporate traces. "Perfect recall" means that every possible world incorporates information how it was generated. "Synchronous" means that traces of different lengths can be distinguished.

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 $PRS(E) = (D_{PRS(E)}, \overset{1}{\underset{\text{prs}}{\sim}}, .. \overset{n}{\underset{\text{prs}}{\sim}}, I_{PRS(E)}, V_{PRS(E)})$, where

- $D_{PRS(E)}$ is the set of all pairs $(wrld, acts)$, where $wrld \in D_E^+$, $acts \in Act^*$, $|wrld| = |acts| + 1$, and $(wrld_j, wrld_{j+1}) \in I_E(acts_j)$ for every $j \in [1..|acts|]$;
- for every $i \in [1..n]$ and for all $(wrld', acts')$, $(wrld'', acts'') \in D_{PRS(E)}$, $(wrld', acts') \overset{i}{\underset{\text{prs}}{\sim}} (wrld'', acts'')$ iff $acts' = acts''$ and $wrld'_j \overset{i}{\sim} wrld''_j$ for every component j;

- for every $a \in Act$ and for all $(wrld', acts'),\ (wrld'', acts'') \in D_{PRS(E)}$, $((wrld', acts'),\ (wrld'', acts'')) \in I_{PRS(E)}(a)$ iff $acts'^\wedge a = acts''$, and $wrld'' = wrld'^\wedge w''$, $(w', w'') \in I_E(a)$, where w' and w'' are the last elements in $wrld'$ and $wrld''$, respectively;
- for every $p \in Prp$ and for every $(wrld, acts) \in D_{PRS(E)}$, $(wrld, acts) \in V_{PRS(E)}(p)$ iff $wrld_{|wrld|} \in V_E(p)$.

In this section we are going to have a close look at the simulation power of the synchronous perfect recall semantics. In particular, we would like to simulate computations of Turing machines in some special class $\mathcal{CL}$ and satisfiability for formulae of the Weak Second-order logic of a single Successor $WS(1)S$.

Tape alphabets of machines in $\mathcal{CL}$ include three special fixed distinguishable symbols L, R, and B: L and R are markers for the left and the right end of the working space, while B is a special disjoint blank symbol for the outer space. Machines in $\mathcal{CL}$ never stay on a tape cell while in work. They never moves to the left of the marker L, but they can move the right marker R cell by cell from left to right as far as required. In the initial configuration the working head is observing the marker L in the leftmost position. In the accepting configuration the working head is observing the marker R in the rightmost position. The following proposition is inspirited by [22]:

Proposition 6. *Let 'next' be a fixed action symbol.*

There exists a PLC_2 formula ϕ such that for every machine $\mathcal{M} \in \mathcal{CL}$ there exists a finite environment E such that for every $m \geq 0$ and every input α for $\mathcal{M}$ there exists a sequence of worlds $wrld$ with $|wrld| = m$ such that

$$\mathcal{M} \textit{ halts on } \alpha \textit{ utilizing } m \textit{ cells iff } (wrld, next^{(m-1)}) \models_{PRS(E)} \phi.$$

The formula ϕ can be constructed in the constant time, the environment E – in time $O(|\mathcal{M}|)$, the sequence $wrld$ – in time $O(|\alpha|)$.

Proof (sketch).

Let $\mathcal{M}$ be a Turing machine in $\mathcal{CL}$ with the tape alphabet Σ, control states Q and a program Π. A configuration $cnfg$ of a Turing machine is a word in the alphabet $\Sigma \bigcup (\Sigma \times Q)$ with a single instance of $\Sigma \times Q$. Let α be an input for $\mathcal{M}$, and $m \geq 0$. Let us begin from a finite environment E for two agents. Let domain D_E be a union of the following sets: Σ, $(\Sigma \times Q)$, $(\Sigma \times \{left, right\})$, and $(\Sigma \times \{left, right\} \times \{next, prev\} \times Q)$.

Let $\overset{1}{\sim}$ be the least equivalence relation such that

$$\sigma \overset{1}{\sim} (\sigma, \mathit{dir}) \overset{1}{\sim} (\sigma, \mathit{dir}, \mathit{next}, q) \text{ and } (\sigma, q) \overset{1}{\sim} (\sigma, \mathit{dir}, \mathit{prev}, q)$$

for every $\sigma \in \Sigma$, every $\mathit{dir} \in \{\mathit{left}, \mathit{right}\}$, and every $q \in Q$.

Let $\overset{2}{\sim}$ be the least equivalence relation such that

$$(\sigma, q) \overset{2}{\sim} (\sigma, \mathit{dir}, \mathit{next}, q)\ ,\ \sigma \overset{2}{\sim} (\sigma, \mathit{dir})\ ,\ \text{and } \sigma'' \overset{2}{\sim} (\sigma', \mathit{dir}, \mathit{prev}, q)$$

for all $\sigma, \sigma', \sigma'' \in \Sigma$ and every $q \in Q$ such that $\big((\sigma', q) \to (\sigma'', ..., \mathit{dir})\big) \in \Pi$.

Let interpretation I_E of a single action symbol $\mathit{next} \in \mathit{Act}$ comprises all pairs of worlds of the following kinds for all $\sigma, \sigma', \sigma'' \in \Sigma$ and all $q, q', q'' \in Q$:

- $B \xrightarrow{\mathit{next}} B$ and $(B, q) \xrightarrow{\mathit{next}} B$,
- $\sigma' \xrightarrow{\mathit{next}} \sigma''$, $\sigma' \xrightarrow{\mathit{next}} (\sigma'', q)$ and $(\sigma', q) \xrightarrow{\mathit{next}} \sigma''$, where $\sigma' \not\equiv B$,
- $(\sigma', \mathit{left}) \xrightarrow{\mathit{next}} (\sigma'', \mathit{left})$ and $(\sigma', \mathit{left}) \xrightarrow{\mathit{next}} (\sigma'', \mathit{left}, \mathit{next}, q)$,
- $(\sigma', \mathit{left}, \mathit{next}, q') \xrightarrow{\mathit{next}} (\sigma'', \mathit{left}, \mathit{prev}, q'')$
  iff $\big((\sigma'', q'') \to (..., q', \mathit{left})\big) \in \Pi$,
- $(\sigma', \mathit{left}, \mathit{prev}, q) \xrightarrow{\mathit{next}} (\sigma'', \mathit{left})$,
- $(\sigma', \mathit{right}) \xrightarrow{\mathit{next}} (\sigma'', \mathit{right})$ and $(\sigma', \mathit{right}) \xrightarrow{\mathit{next}} (\sigma'', \mathit{right}, \mathit{prev}, q)$,
- $(\sigma', \mathit{right}, \mathit{prev}, q') \xrightarrow{\mathit{next}} (\sigma'', \mathit{right}, \mathit{next}, q'')$
  iff $\big((\sigma'\ , q') \to (..., q'', \mathit{right})\big) \in \Pi$,
- $(\sigma', \mathit{right}, \mathit{next}, q) \xrightarrow{\mathit{next}} (\sigma'', \mathit{right})$.

Let valuation V_E of propositional variables IsL, IsR, and IsB be worlds L, R, and B. Let valuation V_E of a propositional variable AtA be a single world (B, accept).

Thus a finite environment E is defined[13]. Now we can define the PLC$_n$ formula ϕ and the sequence of worlds wrld. Let ϕ be $C_{\{1,2\}}(\mathit{AtA} \wedge IsR)$ and let wrld be $L\alpha RB^{(m-|\alpha|-2)}$.

The environment E enjoys the following step-simulation property that holds for all sequences of worlds wrld' and wrld'':

1. if wrld' is a configuration of $\mathcal{M}$, then

   $$(\mathit{wrld}', \mathit{next}^{(m-1)}) \underset{\mathrm{prs}}{\overset{1}{\sim}} (\mathit{wrld}'', \mathit{next}^{(m-1)}) \text{ iff } \mathit{wrld}' = \mathit{wrld}'' \text{ or } \mathit{wrld}''$$

   is a semi-next configuration of $\mathcal{M}$, where all cells have the same marks

[13] Please see Appendix A for an example.

extended with consistent information about the movement direction, the next active cell has information about the next control state, and the active cell remembers the current control state;

2. if $wrld'$ is a configuration of $\mathcal{M}$, then

$$(wrld', next^{(m-1)}) \overset{2}{\underset{\text{prs}}{\sim}} (wrld'', next^{(m-1)}) \text{ iff } wrld' = wrld'' \text{ or } wrld''$$

   is a semi-previous configuration of $\mathcal{M}$, where all cells have the same marks extended with consistent information about the movement direction, the active cell has information about the control state, and the previous active cell remembers the previous control state and the previous mark.

Due to the step-simulation property, we have:

$$\begin{gathered}
\mathcal{M} \text{ halts on } \alpha \text{ utilizing } m \text{ cells}\\
\Updownarrow\\
\text{there exists a finite sequence of configurations } cnfg \text{ such that}\\
\text{every configuration is of length } m,\\
cnfg_1 \text{ is } L\alpha RB^{(m-|\alpha|-2)},\ cnfg_{|cnfg|} \text{ is accepting,}\\
\text{and } cnfg_{(j+1)} \text{ is a consequent of } cnfg_j \text{ for every } j\\
\Updownarrow\\
\text{there exists a finite sequence of sequences of } E \text{ worlds } cnfg \text{ such that}\\
\text{every sequence is of length } m,\\
cnfg_1 \text{ is } L\alpha RB^{(m-|\alpha|-2)},\ cnfg_{|cnfg|} \models_E (AtA \wedge IsR),\\
\text{and } (cnfg_j, next^{(m-1)})(\overset{1}{\underset{\text{prs}}{\sim}} \circ \overset{2}{\underset{\text{prs}}{\sim}})(cnfg_{(j+1)}, next^{(m-1)}) \text{ for every } j\\
\Updownarrow\\
(L\alpha RB^{(m-|\alpha|-2)}, next^{(m-1)}) \models_{PRS(E)} C_{\{1,2\}}(AtA \wedge IsR).
\end{gathered}$$

■

The Weak Second-Order logic of 1 Successor $WS(1)S$ can be defined in different manners [2, 25, 24, 26, 1]. We would like to exploit the following notation. Terms and elementary formulae of $WS(1)S$ are constructed from

- first- and second-order variables $x, y, ...$ and $Z, ...$,
- a constant symbol e and a single monadic functional symbol S,
- binary predicate symbols $=$ and $\in$.

Formulae of $WS(1)S$ are constructed from elementary formulae by means of

- standard propositional connectives $\neg$, $\wedge$, etc.,
- first- and second-order quantifiers $\forall$ and $\exists$.

Natural numbers $\{0, 1, 2, 3, ...\}$ are the model for $WS(1)S$, where e represents 0 and S represents the successor function $\lambda j.(j+1)$. Evaluation is a mapping which assigns a natural number to every first-order variable and a finite set of natural numbers to every second-order variable. Semantics of formulae of $WS(1)S$ in this standard model is defined in terms of satisfiability of formulae on evaluations in a natural way, with first-order variables and quantifiers ranging over natural numbers and second-order variables and quantifiers ranging over finite sets of natural numbers.

The Second-Order logic of 1 Successor $S(1)S$ [2, 25, 24, 26, 1] has the same syntax as $WS(1)S$ and a very similar semantics. The only difference is semantics of the second-order variables and quantifiers: in $S(1)S$ they range over *all* sets of natural numbers instead of *finite* sets in $S(1)S$.

Proposition 7. *Let 'next' be a fixed action symbol.*

For every formula ϕ of $WS(1)S$, there exist a finite environment E and a set of worlds T such that for every $m \geq 0$ there exist

1. *a one-to-one correspondence vt between evaluations in $[0..(m-1)]$ of variables occurring in ϕ and next-traces of length m in E that finish in T,*
2. *a translation tr of subformulae ϕ into formulae of Act-CTL-K_2 with $next \in Act$*

such that for every evaluation ev in $[0..m-1]$ of variables occurring in ϕ and every subformula ψ of ϕ the following holds:

$$ev \models \psi \text{ iff } (vt(ev), next^{(m-1)}) \models_{PRS(E)} tr(\psi).$$

The environment E and the set T can be constructed in time $exp(|\phi|)$, the correspondence vt — in time $O(m \times |\phi|)$, and the translation tr — in time $O(|\phi|)$.

Proof (sketch).

Let ϕ be a formula of $WS(1)S$. We can assume that all bounded variables are pairwise disjoint, i.e., there is no name collisions. We would like to begin from a multiple-agent case and then we hint at simulation of multiple-agents by 2-agents.

n-agent case.

Let us define E and T first, then – vt and tr.

The worlds D_E are vectors of 0, 1 and U with positions corresponding to variables and elementary subformulae in ϕ, but the value U can occur in positions corresponding to the second-order variables only. For every vector

- $(..., \overset{x:}{0}, ...) \overset{next}{\not\rightarrow} (..., \overset{x:}{1}, ...)$ for every first-order variable x;
- $(..., \overset{Z:}{U}, ...) \overset{next}{\not\rightarrow} (..., \overset{Z:}{1}, ...)$, $(..., \overset{Z:}{U}, ...) \overset{next}{\not\rightarrow} (..., \overset{Z:}{0}, ...)$
  $(..., \overset{Z:}{0}, ...) \overset{next}{\not\rightarrow} (..., \overset{Z:}{U}, ...)$ for every second-order variable Z;
- $(..., \overset{(x=y):}{0}, ...) \overset{next}{\not\rightarrow} (..., \overset{(x=y):}{1}, ...)$, $(..., \overset{(x=y):}{1}, ...) \overset{next}{\not\rightarrow} (..., \overset{(x=y):}{0}, ...)$,
  $(..., , \overset{x:}{1}, ... \overset{y:}{1}, ... \overset{(x=y):}{0}, ...) \overset{next}{\not\rightarrow} (..., , \overset{x:}{0}, ... \overset{y:}{0}, ... \overset{(x=y):}{0}, ...)$,
  $(..., , \overset{x:}{1}, ... \overset{y:}{0}, ... \overset{(x=y):}{1}, ...) \overset{next}{\not\rightarrow} (..., , \overset{x:}{?}, ... \overset{y:}{0}, ... \overset{(x=y):}{1}, ...)$,
  $(..., , \overset{x:}{0}, ... \overset{y:}{1}, ... \overset{(x=y):}{1}, ...) \overset{next}{\not\rightarrow} (..., , \overset{x:}{0}, ... \overset{y:}{?}, ... \overset{(x=y):}{1}, ...)$
  for all first-order variables x and y;
- $(..., \overset{(x \in Z):}{0}, ...) \overset{next}{\not\rightarrow} (..., \overset{(x \in Z):}{1}, ...)$, $(..., \overset{(x \in Z):}{1}, ...) \overset{next}{\not\rightarrow} (..., \overset{(x \in Z):}{0}, ...)$,
  $(..., , \overset{x:}{1}, ... \overset{Z:}{1}, ... \overset{(x \in Z):}{0}, ...) \overset{next}{\not\rightarrow} (..., , \overset{x:}{0}, ... \overset{Z:}{?}, ... \overset{(x \in Z):}{0}, ...)$,
  $(..., , \overset{x:}{1}, ... \overset{Z:}{0}, ... \overset{(x \in Z):}{1}, ...) \overset{next}{\not\rightarrow} (..., , \overset{x:}{0}, ... \overset{Z:}{?}, ... \overset{(x \in Z):}{1}, ...)$,
  $(..., , \overset{x:}{1}, ... \overset{Z:}{U}, ... \overset{(x \in Z):}{1}, ...) \overset{next}{\not\rightarrow} (..., , \overset{x:}{?}, ... \overset{Z:}{U}, ... \overset{(x \in Z):}{1}, ...)$
  for every first-order variable x and every second-order variable Z.

Fig. 6. Constraints for interpretation of *next*

vec of this kind and every position *pst*, the value of the correspondent position in *vec* is denoted as vec_{pst}. In particular, let T be a set of worlds which comprises all vectors *vec* such that

- $vec_x = 0$ for every first-order variable x,
- $vec_Z = U$ for every second-order variable Z.

For every variable i which occurs in ϕ, let $\overset{i}{\sim}$ be an equivalence relation such that for all worlds vec' and vec''

$$vec' \overset{i}{\sim} vec''$$

iff

$$vec'_{pst} = vec''_{pst}$$

for every position *pst*

that does not correspond to i or to a subformula with instances of i (i.e., $\overset{i}{\sim}$ can vary a component for i and all components which depend on i).

Let interpretation I_E of a single action symbol *next* comprises all pairs of worlds which meet the prohibition constraints presented in Fig. 6 (where "?" stays for "any value").

For every elementary subformula ξ, let valuation V_E of a propositional variable $ch\xi$ be worlds vec with $vec_\xi = 1$. Let $term$ be a new propositional variable and valuation V_E of this variable be the set T.

Thus the environment E and the set T are defined.

For every $j \geq 0$ and every evaluation ev of variable which occurs in ϕ, let us postulate that

- $ev(x,j) = \begin{cases} 1, \text{ if } ev(x) \geq j \\ 0, \text{ otherwise} \end{cases}$
  for every first-order variable x;
- $ev(Z,j) = \begin{cases} 1, \text{ if } j \in ev(Z) \\ 0, \text{ if } j \not\in ev(Z) \text{ but } k \in ev(Z) \text{ for some } k > j \\ U, \text{ otherwise} \end{cases}$
  for every second-order variable Z;
- $ev(\xi,j) = \begin{cases} 1, \text{ if } ev \models \xi \\ 0, \text{ otherwise} \end{cases}$
  for every elementary subformula ξ.

For every $m \geq 0$, let $vt : ev \mapsto vec_1...vec_m$ be a mapping from evaluations in $[0..(m-1)]$ of variables in ϕ to sequences of worlds $wrld$ of length m: $(wrld_j)_{elm} = ev(elm, j)$ for every $j \in [1..m]$ and every first- or second-order variable or elementary subformula elm. It is straightforward that $vt(ev)$ is a *next*-trace of length m that finishes in the set T for every evaluation ev in $[0..(m-1)]$ of variables in ϕ. Vice versa, for every *next*-trace $wrld$ of length m that finishes in the set T, there exists a unique evaluation ev in $[0..(m-1)]$ of variables in ϕ such that $wrld = vt(ev)$, since $wrld$ encodes ev as follows:

- $(wrld_1)_x...(wrld_m)_x$ is a monadic representation of a value $ev(x) \in [0..(m-1)]$ for every first-order variable x;
- $(wrld_1)_Z...(wrld_m)_Z$ is a ternary[14] representation of the characteristic function of a set $ev(Z) \subseteq [0..(m-1)]$ for every second-order variable Z;
- $(wrld_1)_\xi...(wrld_m)_\xi$ is a constant in $\{0,1\}$ for $ev \models \xi$ and $ev \not\models \xi$ for every elementary subformula ξ.

Thus vt is a one-to-one correspondence.

Let us define a translation tr as follows:

- $tr(\xi) = (term \wedge ch\xi)$ for every elementary subformula ξ;

[14] Here 1 is used for elements and 0 and U are used for non-elements less and, respectively, greater than $max(ev(Z))$.

- $tr(\neg\psi) = \neg(tr(\psi))$, $tr(\psi' \wedge \psi'') = (tr(\psi') \wedge tr(\psi''))$, $tr(\psi' \vee \psi'') = (tr(\psi') \vee tr(\psi''))$ for all subformulae ψ, ψ', and ψ'';
- $tr(\exists x.\psi) = \mathbf{EF}\ S_x\ (term \wedge tr(\psi))$, $tr(\forall x.\psi) = \mathbf{AG}\ K_x\ (term \to tr(\psi))$ for every first-order variable x and every subformula ψ;
- $tr(\exists Z.\psi) = \mathbf{EF}\ S_Z\ (term \wedge tr(\psi))$, $tr(\forall Z.\psi) = \mathbf{AG}\ K_Z\ (term \to tr(\psi))$ for every second-order variable x and every subformula ψ.

The property $ev \models \psi$ iff $(vt(ev), next^{(m-1)}) \models_{PRS(E)} tr(\psi)$ for all $m \geq 0$, evaluation in $[0..(m-1)]$ and subformula ψ can be proved by induction on the structure of a subformula. Inductive steps for elementary subformulae and propositional combinations are straightforward. Inductive steps for quantifiers are similar to each other, so we would like to consider only a second-order case $(\exists Z.\psi)$:

$$\begin{gathered} ev \models (\exists Z.\psi) \\ \Updownarrow \\ ev' \models \psi \text{ for some } ev' \text{ which agrees with } ev \text{ everywhere but } Z \\ \text{inductive} \Updownarrow \text{hypothesis} \\ vt(ev') \models_{PRS(E)} tr(\psi) \text{ for some } ev' \text{ which agrees with } ev \text{ everywhere but } Z \\ \text{definition of} \Updownarrow \text{correspondence } vt \\ vt(ev) \models_{PRS(E)} (\mathbf{EF} K_Z tr(\psi)) \\ \Updownarrow \\ vt(ev) \models_{PRS(E)} tr(\exists Z.\psi). \end{gathered}$$

Thus the n-agent case is over.

2-agent case.

In this case each world has 2 reserved positions for every variable i: the position $v(i)$ for a value and $c(i)$ — for a copy. In general, the value and copy positions can be different, but we are interested essentially in good worlds which contains equal values in the corresponding positions. For every variable i let $Vr(i)$ be a distinct special propositional variable. Every world has a single position for every special propositional variable. A special propositional variable is valid in those worlds where the corresponding position contains 1. An interpretation of a single action symbol $next$ should meet the following additional constraint: if $vec' \xrightarrow{next} vec''$ then for every i

$$vec''_{Vr(i)} = vec'_{Vr(i)} \wedge \begin{pmatrix} \text{for every } j \not\equiv i \\ vec'_{v(j)} = vec'_{c(j)} \end{pmatrix} \wedge \begin{pmatrix} \text{for every } j \not\equiv i \\ vec''_{v(j)} = vec''_{c(j)} \end{pmatrix}$$

i.e., $vec''_{Vr(i)}$ accumulates differences between the value and copy positions for all variables but i in vec' and vec''. There are only 2 agents: agent 1 sees only the copy positions and agent 2 sees only the value positions.

Every step through K_1 changes values of variables, but keeps the copies the same. Checking $Vr(i)$ restricts all changes to changes of values of a variable i. Every step through K_2 changes copies of variables, but keeps the values the same. Checking $Vr(i)$ for all variables i makes all copies equal to values. Thus we can simulate the n-agent case constructions $(K_i ...)$ and $(S_i ...)$ by the 2-agent case constructions $K_1(Vr(i) \to K_2(OK \to ...))$ and $S_1(Vr(i) \wedge S_2(OK \wedge ...))$, where OK is conjunction of $Vr(i)$ for all i. ■

7. MODEL CHECKING OF KNOWLEDGE ACQUISITION

We are going to examine the model checking problem for combined logics EPDL-K$_n$, EPDL-C$_n$, *Act*-CTL-K$_n$, *Act*-CTL-C$_n$, μPLK$_n$, and μPLC$_n$ in perfect recall synchronous environments generated from finite environments in the following conditions: what is complexity of the set $CHECK(L) \equiv \{(E,\ (wrld, acts),\ \phi) :$ E is a finite environment, $(wrld, acts) \in D_{PRS(E)}, \phi$ is a formula of L, and $(wrld, acts) \models_{PRS(E)} \phi\}$, where L is a particular combined logic?

Let us discuss parameters used to measure the model checking complexity. We can assume that presentation of every world has some fixed complexity, as well as presentation of every action symbol. If $E = (D_E, \overset{1}{\sim}, \ldots \overset{n}{\sim}, I_E, V_E)$ is a finite background environment presented as a finite graph, then let d_E and r_E be the number of worlds in D_E and the number of pairs of worlds in I_E; let m_E be an overall complexity $(d_E + r_E)$. If $(wrld, acts) \in D_{PRS(E)}$, then let $l_{(wrld,acts)}$ be $|wrld| = |acts| + 1$. If $(wrld, acts)$ is implicit, then we would like to use these parameters without subscripts, i.e., just l. If ϕ is a formula, then let f_ϕ be the size of ϕ. A natural complexity measure for triples $(E\ ,\ (wrld, acts)\ ,\ \phi)$, where E is a finite environment, $(wrld, acts) \in D_{PRS(E)}$, and ϕ is a formula, is $(m_E + l_{(wrld,acts)} + f_\phi)$.

Proposition 8. *For all $n > 1$ and Act $\neq \emptyset$, $CHECK$(EPDL-C$_n$) is $PSPACE$-complete.*

Proof. By proposition 6, every polynomial computation of every Turing machine $\mathcal{M}$ can be represented as a model checking problem for some fixed formula ϕ of PLC$_2$ in an environment E which can be constructed in time $O(|\mathcal{M}|)$ on some $(wrld, acts) \in D_{PRS(E)}$ of a polynomial length. It implies that $CHECK$(EPDL-C$_n$) is $PSPACE$-hard.

Let us prove that $CHECK$(EPDL-C$_n$) is in $PSPACE$. Without loss of generality it is possible to consider only the so-called *normal* formulae, i.e., the formulae with negation applied only to propositional variables, due to classical DeMorgan rules and standard modal logic equivalencies $\neg([...]...) \leftrightarrow \langle...\rangle(\neg...)$, $\neg(\langle...\rangle...) \leftrightarrow [...](\neg...)$, $\neg(C_{...}...) \leftrightarrow J_{...}(\neg...)$, and $\neg(J_{...}...) \leftrightarrow C_{...}(\neg...)$. Let us define an alternating Turing machine $\mathcal{ATM}$ with its input alphabet consisting of triples $(E, (wrld, acts), \phi)$, where E is an environment, $(wrld, acts) \in D_{PRS(E)}$, and ϕ is a normal formula of EPDL-C$_n$:

- $\mathcal{ATM}$ on $(E, (wrld, acts), (\neg)p)$ should check, whether $(\neg)p$ holds in the last world of the sequence $wrld$ (where $p \in Prp$);
- $\mathcal{ATM}$ on $(E, (wrld, acts), (\phi' \frac{\wedge}{\vee} \phi''))$ should enter an $\frac{universal}{existential}$ control state where it initiates two independent computations on $(E, (wrld, acts), \phi')$ and on $(E, (wrld, acts), \phi'')$;
- $\mathcal{ATM}$ on $(E, (wrld, acts), (\frac{[a]}{\langle a\rangle}\phi'))$ where $a \in Act$, should enter an $\frac{universal}{existential}$ control state, where it initiates independent computations on $(E, ((wrld^\wedge w), (acts^\wedge a)), \phi')$ for every required world w;
- $\mathcal{ATM}$ on $(E, (wrld, acts), (\frac{C_G}{J_G}\phi'))$, where $G \subseteq [1..n]$, should enter an $\frac{universal}{existential}$ control state where it initiates independent computations on $(E, (wrld', acts), \phi')$ for every possible sequence of worlds $wrld'$, such that $(wrld', acts) \overset{G}{\underset{\text{prs}}{\sim}} (wrld, acts)$.

For every environment E, every $(wrld, acts) \in D_{PRS(E)}$, and every EPDL-C$_n$ normal formula ϕ, the following two statements are equivalent:

- $(wrld, acts) \models_{PRS(E)} \phi$;
- $\mathcal{ATM}$ accepts $(E, (wrld, acts), \phi)$.

Computations of $\mathcal{ATM}$ on an input $(E, (wrld, acts), \phi)$ have $O(|\phi|)$ alternations. They utilize a polynomial space, since the most complicated checking along these computations $(wrld', acts') \overset{G}{\underset{\text{prs}}{\sim}} (wrld'', acts'')$ can be examined according to the strategy "divide and conquer"[15]. It is known that alternating computations which utilizes the space s and k alternations can be simulated by deterministic computations within the space $O(k \times s + s^2)$ [5]. It implies that $CHECK$(EPDL-C$_n$) is in $PSPACE$. ■

Proposition 9. *For all $n > 1$ and $Act \neq \emptyset$, $CHECK$(Act-CTL-K$_n$) is decidable with the upper and lower bounds $2^{2^{\cdot^{\cdot^{2}}}}\Big\}^{O(t)}$, where t is the overall*

[15] For every finite oriented graph (N, R) with nodes N and edges R, for all nodes n' and n'', the problem whether there is a path from n' to n'' can be solved in time $O(|E| \times lg(|N|))$ [19].

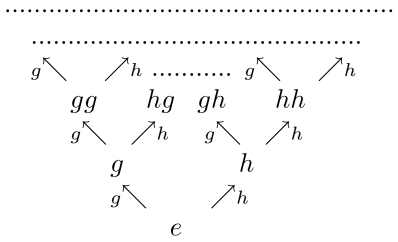

Fig. 7. The model for CLE with two functional symbols g and h

complexity of an input triple.

Proof. It is known that the Weak Second-Order logic of 1 Successor $WS(1)S$ is decidable with a non-elementary lower bound $\left.2^{2^{\cdot^{\cdot^{2}}}}\right\}^{O(f)}$, where f is the size of an input formula [24] (see also [25, 26, 1]). By proposition 7, the decidability problem for $WS(1)S$ can be reformulated as a model checking problem for CTL-K$_2$ in perfect recall synchronous settings. Complexity of this encoding is exponential. These arguments imply a non-elementary lower bound for $CHECK$(*Act*-CTL-K$_n$).

A principle decidability of $CHECK$(*Act*-CTL-K$_n$) with a non-elementary upper bound $\left.2^{2^{\cdot^{\cdot^{2}}}}\right\}^{O(t)}$ is based on reduction of *Act*-CTL-K$_n$ to the Chain Logic with Equal-length predicate CLE [28].

The syntax of CLE is very similar to the syntax of $WS(1)S$, but there are several monadic functional symbols f_1,...f_k instead of a single f and two extra binary predicate symbols E and $\preceq$. The model for CLE is just a full ordered infinite k-ary tree whose nodes are first-order terms without free variables generated from the constant symbol e and functional symbols $f_1, ...f_k$. (For example, Fig. 7 depicts the model for two functional symbols g and h.) A *chain* is a subset of this tree which is totally ordered by the prefix relation. A *path* is a maximal chain with respect to the set inclusion. Semantics of formulae of CLE in this standard model is defined similarly to semantics of formulae of $WS(1)S$ and $S(1)S$, but

- the second-order variables and quantifiers range only over chains in contrast to Rabin logic where they range over arbitrary subsets of the tree;
- the predicate symbol $\preceq$ is interpreted by the first-order binary predicate $\{(t, t') : t \text{ is a prefix of } t'\}$;
- the Equal-length predicate symbol E is interpreted by a first-order

binary equal-length predicate $\{(t, t') : |t| = |t'|\}$.

Let $E = (D_E, \overset{1}{\sim}, .. \overset{n}{\sim}, I_E, V_E)$ be a finite environment. We would like to consider every world of this environment as a monadic functional symbol. We would also like to consider every action symbol as a monadic functional symbol too. (Thus, we assume k to be $(|D_E| + |Act|)$.) We are going to encode every finite sequence of worlds $w_1..w_s$ by a term $w_s(..w_1(e))$ and every finite sequence of action symbols $a_1..a_t$ – by a term $a_t(..a_1(e))$.

Let us consider the following property: a pair of terms $w_s(..w_1(e))$ and $a_t(..a_1(e))$ encodes an element $(w_1..w_s, a_1..a_t)$ of $D_{PRS(E)}$. This property is expressible in CLE, and we would like to denote by $IN_{PRS(E)}(x, y)$ a formula of CLE which expresses this property for the values of x and y.

Let T be the following "semi-formal" algorithm:

- $T(E, (w_1..w_s, a_1..a_t), p) =$
  $= IN_{PRS(E)}(w_s(..w_1(e)), a_t(..a_1(e))) \ \wedge \ w_s \in V_E(p)$,
  where $p \in Prp$;
- $T(E, (w_1..w_s, a_1..a_t), (\neg\phi)) =$
  $= IN_{PRS(E)}(w_s(..w_1(e)), a_t(..a_1(e))) \ \wedge \ \neg T(E, (w_1..w_s, a_1..a_t), \phi)$;
- $T(E, (w_1..w_s, a_1..a_t), (\phi' \frac{\wedge}{\vee} \phi'')) =$
  $= T(E, (w_1..w_s, a_1..a_t), \phi') \frac{\wedge}{\vee} T(E, (w_1..w_s, a_1..a_t), \phi'')$;
- $T(E, (w_1..w_s, a_1..a_t), (\frac{\mathbf{AX}^a}{\mathbf{EX}^a} \phi)) =$
  $= \frac{\bigwedge_{(w_s, w) \in I_E(a)}}{\bigvee_{(w_s, w) \in I_E(a)}} T(E, (w_1..w_s w, a_1..a_t a), \phi)$,
  where $a \in Act$;
- $T(E, (w_1..w_s, a_1..a_t), \frac{\mathbf{A}}{\mathbf{E}}(\phi' \mathbf{U}^a \phi'')) =$
  $= IN_{PRS(E)}(w_s(..w_1(e)), a_t(..a_1(e))) \wedge$
  $\frac{\forall}{\exists}$ path $WRLD$ which starts with $w_s(..w_1(e))$
  $\frac{\forall}{\exists}$ path $ACTS$ which consists of terms $a^k(a_t(..a_1(e)))$
  $\exists x \in WRLD \ \exists y \in ACTS \ :$
  $(IN_{PRS(E)}(x, y) \ \wedge \ T(E, (x, y), \phi'') \ \wedge$
  $\forall u \prec x \ \forall v \prec y \ :$
  $(w_s(..w_1(e)) \preceq u \wedge a_t(..(a_1(e)) \preceq v \wedge IN_{PRS(E)}(u, v) \ \rightarrow$
  $\rightarrow T(E, (x, y), \phi')))$;
- $T(E, (w_1..w_s, a_1..a_t), (\frac{K_i}{S_i} \phi)) \ = \ IN_{PRS(E)}(w_s(..w_1(e)), a_t..a_1(e))) \ \wedge$
  $\wedge \ \frac{\bigwedge_{u_1 \overset{i}{\sim} w_1}}{\bigvee_{u_1 \overset{i}{\sim} w_1}} .. \frac{\bigwedge_{u_s \overset{i}{\sim} w_s}}{\bigvee_{u_s \overset{i}{\sim} w_s}} (IN_{PRS(E)}(u_s(..u_1(e)), a_t(..a_1(e))) \ \frac{\rightarrow}{\wedge}$
  $\frac{\rightarrow}{\wedge} \ T(E, (u_1..u_s, a_1..a_t), \phi))$, where $i \in [1..n]$.

This algorithm has a polynomial complexity. For all environments E, finite sequences of words $wrld$ and actions $acts$, and the Act-CTL-K$_n$ formula ϕ, the following two statements are equivalent:

- $(wrld, acts) \in D_{PRS(E)}$ and $(wrld, acts) \models_{PRS(E)} \phi$;
- $T(E, (wrld, acts), \phi)$ is a valid formula of CLE.

The following facts are proved in [28]: CLE and $S(1)S$ are mutually interpretable; CLE is non-elementary decidable with the upper bound $2^{2^{\cdot^{\cdot^{2}}}}\Big\}^{O(f)}$. These arguments implies an upper bound for $CHECK(Act\text{-CTL-K}_n)$. ■

Proposition 10. *$CHECK(Act\text{-}CTL\text{-}C_n)$, $CHECK(\mu PLK_n)$, and $CHECK(\mu PLC_n)$ are undecidable for all $n > 1$ and $Act \neq \emptyset$.*

Proof. In accordance with proposition 6, every computation of every Turing machine $\mathcal{M}$ which exploits a fixed space can be presented as a model checking problem for some fixed formula ϕ of PLC$_2$. Hence, the halting problem for Turing machines can be represented as a model checking problem for the CTL-C$_2$ formula $\mathbf{EF}\phi$. It implies undecidability of $CHECK(Act\text{-CTL-C}_n)$. In accordance with proposition 1, it implies also undecidability of $CHECK(\mu\text{PLK}_n)$ and $CHECK(\mu\text{PLC}_n)$. ■

8. FORMULAE WITH BOUNDED KNOWLEDGE DEPTH

The above proposition 9 establishes a non-elementary lower bound for the model checking problem for *Act*-CTL-K in the synchronous environments with perfect recall generated from finite background environments. It exploits the overall complexity of the problem inputs and does not distinguish a complexity impact of every input. In this section we would like to develop some techniques for evaluation of a contribution of every input to this non-elementary overall complexity. First, let us redefine the knowledge depth for formulae of *Act*-CTL-K$_n$ and related sublogics *Act*-CTL-K$_n^k$, $k \geq 0$, with a bounded knowledge depth, then redefine k-trees and revise some related results [22], and finally let us prove that these trees are abstraction of the PRS-environments with respect to formulae of *Act*-CTL-K$_n^k$. For simplicity of presentation, let us fix a finite environment $E = (D_E, \overset{1}{\sim}, .. \overset{n}{\sim}, I_E, V_E)$ in this section. Thus the corresponding synchronous environment with perfect recall $PRS(E) = (D_{PRS(E)}, \underset{\text{prs}}{\overset{1}{\sim}}, .. \underset{\text{prs}}{\overset{n}{\sim}}, I_{PRS(E)}, V_{PRS(E)})$ is also fixed and we would like to refer to it simply as PRS.

The knowledge depth of a formula is the maximal nesting of knowledge operators in that formula. For example, $depth(K_1(\mathbf{EX}K_2(q \wedge K_2 r))) = 3$.

Let *Act*-CTL-K$_n^k$ be sublogics of *Act*-CTL-K$_n$ with a bounded knowledge depth $k \geq 0$. Naturally, *Act*-CTL-K$_n$ $= \bigcup_{k \geq 0}$ *Act*-CTL-K$_n^k$.

For every integer $k \geq 0$ we define by mutual recursion the set $\mathcal{T}_k$ of *k-trees over E*, and the set $\mathcal{F}_k$ of *forests of k-trees over E*. Let $\mathcal{T}_0$ be the set of all tuples of the form $(w, \emptyset, ...\emptyset)$, where w is a world and the number of copies of the empty set $\emptyset$ is equal to the number of agents n. Once $\mathcal{T}_k$ has been defined, let $\mathcal{F}_k$ as the set of all subsets of $\mathcal{T}_k$. Now, define $\mathcal{T}_{k+1}$ as a the set of all tuples of the form $(w, U_1, ...U_n)$, 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 labelled by worlds of the environment E and edges are labeled by agents. In a tuple $(w, U_1, ...U_n)$, 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 $(w, \emptyset, ...\emptyset)$ 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.

We would like also to remark that trees defined above are a little bit different from the trees defined in [22]. For every $i \in \{1..n\}$, a k-tree (≥ 1) a-la [22] 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 state and hence, roughly speaking, are idempotent due to 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.

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

- *the number of k-trees over E C_k 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:
$C_{j+1} \leq |\{ (w, U_1, ...U_n) : w \in D,\ U_1, ...U_n \in \mathcal{P}(\mathcal{T}_j) \}| = d \times (2^{C_j})^n =$
$= d \times 2^{n \times C_j} \overset{ind}{\leq} exp(n \times d, (j+1))/n$.
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 $(wrld, acts)$ be a world of $PRS(E)$. Knowledge available in this world can be represented as an infinite sequence

$$tree_0(wrld, acts)...tree_k(wrld, acts)...,$$

where each $tree_k(wrld, acts)$, $k \geq 0$, is a k-tree defined as follows.
Let $tree_0(wrld, acts)$ be $(wrld_{|wrld|}, \emptyset, ...\emptyset)$, and for every $k \geq 0$ let $tree_{k+1}(wrld,\ acts)$ be
$$\Big(wrld_{|wrld|},\ \{tree_k(wrld', acts') : (wrld', acts') \overset{1}{\underset{\text{prs}}{\sim}} (wrld, acts)\},$$
$$...\{tree_k(wrld', acts') : (wrld', acts') \overset{n}{\underset{\text{prs}}{\sim}} (wrld, acts)\} \Big).$$

Let us define some knowledge update functions for k-trees. Similar functions has been used in [22] to provide an algorithm for model checking problem for formulae of PLK_n in synchronous environments with perfect recall.

Let D_E and I_E be a domain and interpretation of relation symbols from $Acts$ in the environment E. For every number $k \geq 0$, $act \in Acts$ and $i \in \{1..n\}$, the functions $G_k^{act} : \mathcal{T}_k \times D_E \to \mathcal{T}_k$ and $H_{k,i}^{act} : \mathcal{F}_k \times D_E \to \mathcal{F}_k$, are defined by induction on k and mutual recursion. Let

$$G_0^{act}(tr, wrld) = (wrld, \emptyset, ...\emptyset) \text{ iff } (root(tr), wrld) \in I_E(act).$$

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

$$(\ wrld'\ , H_{k,1}^{act}[U_1, wrld'],\ ...\ H_{k,n}^{act}[U_n, wrld']\ ) \text{ iff } (wrld, wrld') \in I_E(act).$$

The following proposition is inspirited by [22]:

Proposition 12. *For every $k \geq 0$, every $act \in Act$, every finite environment E, every $(wrlds, acts) \in D_{PRS(E)}$, every $wrld \in D_E$, and every $act \in Act$, the following incremental knowledge update property holds:*
$tree_k((wrlds, acts)^\wedge\{(wrld, act)\}) \ = \ G_k^{act}(tree_k(wrlds, acts),\ wrld)$.

Proof (sketch) 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 $tree_{k+1}(ws, as) = (w, U_1, ...U_n)$ and $(w, w') \in I_E(a)$. Then

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

Every $G_k^a(tr, w'')$ 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,

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

This proves the inclusion

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

The backward inclusion

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

is similar. ■

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 formulae of Act-CTL-K$_n$ to the formulae of Act^{+n}-CTL is simple: just replace every instance of K_i and S_i by corresponding $\mathbf{AX}^i$ and $\mathbf{EX}^i$, 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 the models for Act^{+n}-CTL: the environment $E = (D_E, \overset{1}{\sim}, \ldots \overset{n}{\sim}, I_E, V_E)$ can be represented naturally as a model $E^{+n} = (D_E, I_E^{+n}, V_E)$, where I_E^{+n} is equal to I_E on action symbols, but $I_E^{+n}(i) = \overset{i}{\sim}$ for every agent $i \in [1..n]$. The following proposition is straightforward:

Proposition 13. *$E(\phi) = E^{+n}(\phi^{+n})$ 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$_n$.*

The above model $PRS(E)^{+n}$ 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. 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;
- $I_{TR_k(E)}(act) = \{(tr', tr'') \in D_{TR_k(E)} :$
  $tr'' = G_j^{act}(tr', wrld)$ for some $j \in [0..k]$ and some $wrld \in D_E$ $\}$
  for every $act \in Act$;
  $I_{TR_k(E)}(i) = \{(tr', tr'') \in D_{TR_k(E)} :$
  $tr'' \in U_i$ and $tr' = (wrld, U_1, \ldots U_n)$ for some $wrld \in D_E$ $\}$
  for every $i \in [1..n]$;
- $V_{TR_k(E)}(p) = \{tr : root(tr) \in V_E(p)\}$ for every $p \in Prp$.

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 a natural manner (i.e., element-wise). 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 $\{(wrlds, acts) : tree_k(wrlds, acts) \in TR\}$. The following proposition can be proved by induction on the formula structure with the help of proposition 12.

Proposition 14. *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 15. *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 the formulae of Act^{+n}-CTL which correspond to the formulae 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.*

In accordance with proposition 2, the complexity of model checking of an *Act*-CTL formula in a finite model is $O(m \times f)$, where m is an overall model complexity and f is a formula complexity. But the cited proposition has been proved under the assumption 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 11) 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 2, 11 and 15 lead to the following proposition.

Proposition 16. *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 defined by induction as follows: $exp(a, 0) = a$ and $exp(a, b+1) = a \times 2^{exp(a,b)}$.

The above propositions 8, 9, 10 and 16 lead to the following theorem.

Theorem 3. *For all $n > 1$ and $Act \neq \emptyset$, the model checking problem for synchronous finitely generated environments with perfect recall*

- *is PSPACE-complete for EPDL-C_n;*
- *is decidable for Act-CTL-K_n with non-elementary upper and lower bounds (which linearly depend on the formula size and non-elementarily depend on the number of states, agents and knowledge depth);*
- *is undecidable for Act-CTL-C_n, μPLK_n, and μPLC_n.*

9. RELATED PAPERS AND CONCLUSION

The paper has studied, from the theoretical point of view, the algorithmic problems for several combinations of propositional program logics. It focuses on the model checking problem in synchronous perfect recall settings for logics which combine knowledge, actions, and fixpoints. It contributes to model checking research and extends the results of the paper [23], where the model checking problem in synchronous perfect recall settings has been examined for fusions of logics PLK and PLC with the Propositional Logic of Linear Time (PLLT). It has been proven in the cited paper [23] that the problem is

- undecidable for PLLT-C$_n$;
- non-elementarily decidable for PLLT-K$_n$,
- $PSPACE$-complete for UNTIL-free PLLT-C$_n$.

Paper [23] has suggested a tree-like data structure for model checking of linear time and knowledge with bounded nesting. This data structure is very convenient for representation of knowledge evolution and update. It comprises the trees whose depth is equal to knowledge nesting [22]. Paper [23] has demonstrated that the model checking problem for PLLT-K$_n$ in the synchronous perfect recall semantics can be reduced to the problem of emptiness of a Büchi automata whose inputs are infinite sequences of these trees. We develop a similar data structure but exploit abstraction and reduction to the model checking problem for a variant of CTL and models where states are trees.

Another related problem is: whether automatic model checking is feasible for PLLT-K$_n$ and *Act*-CTL-K$_n$? Some experience with tree-like data structures (which are similar to the data structure mentioned in the previous paragraph) is reported in [14]. In this experimental research,

- input finite environments are specified in Multi Agent System Language,
- input PLLT-K$_n$ formulae should not have negative/positive instances of knowledge modalities K_i/S_i for any agent $i \in [1..n]$,
- a model checking engine is a finite-state PLLT model checker SMV [4, 6].

Another related paper is [16]. It has studied the decidability problem for combinations of temporal logics PLLT and CTL with logics PLK and PLC in synchronous perfect recall and forgetful settings. In particular, it has demonstrated completeness of the problem in the following classes of complexity:

	PRS	FAS
CTL-C$_n$, $n \geq 2$	Π^1_1	$EXPTIME$
CTL-K$_n$, $n \geq 2$	nonelementary time	$EXPTIME$
CTL-K$_1$ = CTL-C$_1$	doubly-exponential time	$EXPTIME$

Our paper extends the above table on some other combined logics in forgetful asynchronous settings.

Finally, we would like also to point out that the computer science community comes to better understanding of importance of research in propositional program logics on traces. For example, several logics of these kinds have been studied in the context of static analysis [9]. The most powerful of these new logics is the so called reversible fixpoint calculus ($\overleftarrow{\mu}$C). Roughly speaking, this calculus is a fusion of the propositional μ-Calculus with the Propositional Logic of Knowledge for a single agent. It is defined on two-way (backward/forward or past/future) infinite traces in forgetful asynchronous settings. We hope that the above characterization of $\overleftarrow{\mu}$C can be formalized and we contribute to the study of algorithmic problems of these new logics.

REFERENCES

1. **Börger E., Grädel E., Gurevich Yu.** The Classical Decision Problem. — Berlin a.o.: Springer Verlag, 1997.
2. **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.
3. **Bull R.A., Segerberg K.** Basic Modal Logic // Handbook of Philosophical Logic. Vol.II. — Reidel Publishing Company, 1984 (1-st ed.), Kluwer Academic Publishers, 1994 (2-nd ed.). — P. 1–88.
4. **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. — Vol. 98, N 2. — P. 142–170.
5. **Chandra A.K., Kozen D.C., Stockmeyer L.J.** Alternation // J. ACM. — 1981. — Vol. 28, N 1. — P. 114–133.
6. **Clarke E.M., Grumberg O., Peled D.** Model Checking. — London: MIT Press, 1999.
7. **Cleaveland R., Klain M., Steffen B.** Faster Model-Checking for μ-Calculus // Lect. Notes Comput. Sci. — 1993. — Vol. 663. — P. 410–422.
8. **Cousot P., Cousot R.** Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints // Proc. of 4-th PoPL. — ACM Press, 1977. — P. 238–252.
9. **Cousot P., Cousot R.** Temporal Abstract Interpretation // Proc. of 27-th PoPL. — ACM Press, 2000. — P. 12–25.
10. **Emerson E.A.** Temporal and Modal Logic // Handbook of Theoretical Computer Science. Vol.B. — Elsevier, London: The MIT Press, 1990. — P. 995–1072.

11. **Emerson E.A., Jutla C.S., Sistla A.P.** On model-checking for fragments of Mu-Calculus // Lect. Notes Comput. Sci. — 1993. — Vol. 697. — P. 385–396.
12. **Fagin R., Halpern J.Y., Moses Y., Vardi M.Y.** Reasoning about Knowledge. — London: MIT Press, 1995.
13. **Fisher M.J. Ladner R.E.** Propositional dynamic logic of regular programs // J. Comput. System Sci. — 1979. — Vol. 18, N 2. — P. 194–211.
14. **Garanina N.O.** Experiments with Model Checking Knowledge and Time // Ms. Thesis. Mechanics and Mathematics Department of Novosibirsk State University, 2001 (in Russian).
15. **Harel D.** First-Order Dynamic Logic // Lect. Notes Comput. Sci. — 1979. — Vol.68.
16. **Halpern J.Y., Vardi M.Y.** The complexity of Reasoning About Knowledge and Time // Proc. of Symp. on STOC. — 1986. — P. 304–315.
17. **Harel D.** Dynamic Logic // Handbook of Philosophical Logic. Vol.II. — Kluwer Academic Publishers, 1994 (2-nd ed.). — P. 498–604.
18. **Harel D., Kozen D., and Tiuryn J.** Dynamic Logic. — London: MIT Press, 2000.
19. **Immerman I.** Descriptive Complexity. Graduate Texts in Computer Science. — Berlin a.o.: Springer Verlag, 1999.
20. **Kozen D.** Results on the Propositional Mu-Calculus // Theor. Comput. Sci. — 1983. — Vol. 27, N 3. — P. 333–354.
21. **Kozen D., Tiuryn J.** Logics of Programs // Handbook of Theoretical Computer Science. Vol.B. — Elsevier, London: The MIT Press, 1990. — P. 789–840.
22. **van der Meyden R.** Common Knowledge and Update in Finite Environments // Information and Computation. — 1998. — Vol. 140, N 2. — P. 115–157.
23. **van der Meyden R., Shilov N.V.** Model Checking Knowledge and Time in Systems with Perfect Recall // Lect. Notes Comput. Sci. — 1999. — Vol. 1738. — P. 432–445.
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. — Vol. 141. — P. 1–35.
26. **Rabin M.O.** Decidable Theories // Handbook of Mathematical Logic / Ed. by J. Barwise, H. J. Keisler. — North-Holland Pub. Co., 1977. — P. 595–630.
27. **Tarski A.** A lattice-theoretical fixpoint theorem and its applications // Pacific J. Math. — 1955. — Vol. 5. — P. 285–309.
28. **Thomas W.** Infinite trees and automaton-definable relations over ω-words // Theor. Comput. Sci. — 1992. — Vol. 103. — P. 143–159.
29. **Vardi M.Y.** Reasoning about the past with two-way automata // Lect. Notes Comput. Sci. — 1998. — Vol. 1443. — P. 628–641.

A. AN EXAMPLE OF TURING MACHINE

In this section, we give an example of building a finite environment for Turing machine to illustrate proposition 6. Let Turing machine $MT = \langle\ \Sigma, Q, q_i, q_f, \Pi\ \rangle$ duplicate the number of "1" located between symbols L and R. The alphabet of the MT is $\Sigma = \{1, 0, L, R, B\}$, control states are $Q = \{q_i, q_f, q_1, q_2, q_3, q_4, q_5, q_6\}$, and the program is Π:

$$\begin{array}{lll}
L, q_i \rightarrow q_1, right; & 1, q_1 \rightarrow 0, q_1, right; & R, q_1 \rightarrow 1, q_2, right; \\
B, q_2 \rightarrow R, q_3, left; & 1, q_3 \rightarrow q_3, left; & 0, q_3 \rightarrow 1, q_4, left; \\
0, q_4 \rightarrow q_5, right; & 1, q_5 \rightarrow q_5, right; & R, q_5 \rightarrow 1, q_2, right; \\
L, q_4 \rightarrow q_6, right; & 1, q_6 \rightarrow q_6, right; & B, q_6 \rightarrow q_f, left.
\end{array}$$

Look at the several first steps of run when MT duplicates 2 (in this case, number m from proposition 6 is equal to 7):

$$\alpha = L11RB \longrightarrow^*_{MT} \beta = L1111RB$$
$$\overset{q_i}{L}\ 11RB \xrightarrow{1} L\ \overset{q_1}{1}\ 1RB \xrightarrow{2} L0\ \overset{q_1}{1}\ RB \xrightarrow{3}$$
$$L00\ \overset{q_1}{R}\ B \xrightarrow{4} L001\ \overset{q_2}{B}\ B \xrightarrow{5} L001\ \overset{q_3}{R}\ B \cdots$$

By steps 4 and 5 of this run, we demonstrate that the step-simulation property holds in moving of the working head both to the right and to the left, i.e., $cnfg_i \xrightarrow{i} cnfg_{i+1}$ iff $(cnfg_i, next^6)(\overset{1}{\sim} \circ \overset{2}{\sim})(cnfg_{(i+1)}, next^6)$, $i = 4, 5$.

Fig. 8 shows that the first and the last lines are consecutive configurations of Turing machine at step 4. And simultaneously they are traces of

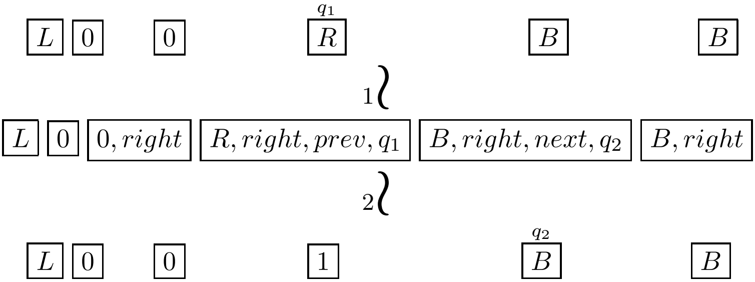

Fig. 8. $L00\ \overset{q_1}{R}\ B \xrightarrow{4} L001\ \overset{q_2}{B}\ B$

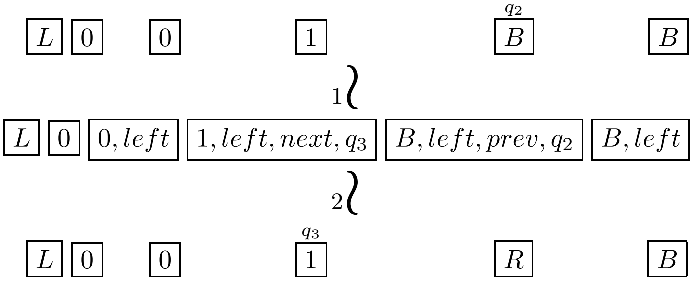

Fig. 9. $L001\overset{q_2}{B}\,B \xrightarrow{5} L001\overset{q_3}{R}\,B$

the induced environment that begins with the letter L. We need to show that there exists another trace such that

- the first agent can not distinguish between this trace and the first line;
- the second agent can not distinguish between this trace and the last line.

The middle line is the trace with the required properties. Indeed, it is a trace by the definition of the action *next* and all its elements are indistiguishable from proper elements of the first (by the first agent) and of the last (by the second agent) line (by the definition of indistiguishability relations). Notice that the first agent knows neither the direction of the move, nor the next state of the machine and the second agent knows neither the direction of the move, nor the previous symbol on the tape, nor the previous state of the machine. All these reasonings are suitable for step 5 represented on Fig.9.

Н. В. Шилов, Н. О. Гаранина

КОМБИНИРОВАНИЕ ЗНАНИЙ И НЕПОДВИЖНЫХ ТОЧЕК

Препринт
98

Рукопись поступила в редакцию 11.04.2002
Рецензент Ф. А. Мурзин
Редактор А. А. Шелухина

Подписано в печать 11.07.2002
Формат бумаги 60×84 1/16 Объем 2,9 уч.-изд.л., 3,1 п.л.
Тираж 50 экз.

НФ ООО ИПО "Эмари" РИЦ, 630090, г. Новосибирск, пр. Акад. Лаврентьева, 6