On Expressive and Model Checking Power of Propositional Program Logics (extended abstract of a short talk)

N. V. Shilov[1,2], K. Yi[1]

[1] Korean Advanced Institute of Science and Technology, Taejon 305-701, Kusong-dong Yusong-gu 373-1, Korea (Republic of Korea),
e-mail: {shilov, kwang}@ropas.kaist.ac.kr
[2] Institute of Informatics Systems, Russian Academy of Sciences, Siberian Division, 6, Lavrent'ev ave., 630090, Novosibirsk, Russia,
e-mail: shilov@iis.nsk.su

Abstract. We examine when a model checker for a propositional program logic can be used for checking another propositional program logic in spite of lack of expressive power of the first logic. We prove that (1) a branching time Computation Tree Logic CTL, (2) the propositional μ-Calculus of D. Kozen μC , and (3) the second-order propositional program logic 2M of C. Stirling enjoy the equal model checking power in spite of difference in their expressive powers CTL $<$ μC $<$ 2M: every listed logic has a formula such that every model checker for this particular formula for models in a class closed w.r.t. finite models, Cartesian products and power-sets can be reused for checking all formulae of these logics in all models in this class. We also suggest a new second-order propositional program logic SOEPDL and demonstrate that this logic is more expressive than 2M, is as expressive as the Second order Logic of monadic Successors of M. Rabin (**S(n)S**-Logic), but still enjoys equal model checking power with CTL, μC and 2M (in the same settings as above).

1 CTL and μC vs. Second-Order Logics

The propositional μ-Calculus of D. Kozen (μC) [10, 11] is a powerful propositional program logic with fixpoints. In particular, a very popular with model checking community state-based temporal Computation Tree Logic (CTL) [7, 4, 5] is interpretable in μC . It is almost a folklore that CTL is less expressive than μC . Nevertheless, expressibility problem rises every time when a particular property is concerned: whether this property can be expressed in CTL, in μC , or both logics are inadequate.

For example, paper [1] reports a progress in experiments with symbolic data representation by means of Binary Decision Diagrams (BDD) for solving some board games. In this research positions and moves are represented by BDD, an existence of winning strategy is expressed as a simple μC formula. Then a new model checking tool for a fragment of μC and finite models presented by BDDs has been applied. A natural question rises: why these experiments have exploited a new symbolic model checking tool instead of utilizing a very popular and reliable symbolic model checker for CTL? A natural move for exploiting a model checker of this kind for solving board games is to try to express an existence of winning strategy in finite games in terms of CTL formulae and then to experiment with these formulae, games presented by BDDs and the model checker. But this move is doomed to fail as follows from the game-theoretic arguments below.

There are different formalisms for games. We would like the following: a game (of two plays A and B) (with terminal positions) is tuple $(P_A, P_B, M_A, M_B, W_A, W_B)$, where

- $P_A \cap P_B = \emptyset$ are nonempty sets of positions,
- $M_A \subseteq (P_A \setminus (W_A \cup W_B)) \times (P_A \cup P_B)$,
  $M_B \subseteq (P_B \setminus (W_A \cup W_B)) \times (P_A \cup P_B)$
  are moves of A and B respectively,
- $W_A, W_B \subseteq (P_A \cup P_B)$ are sets of winning positions for A and B.

The game is said to be finite iff P_A and P_B are finite. A session of the game is a maximal sequence of positions $s_0,\ ...s_n,\ ...$ such that $(s_i, s_{i+1}) \in (M_A \cup M_B)$ for all consequentive positions $s_i, s_{i+1} \in (s_0,\ ...s_n,\ ...)$. A player $C \in \{A, B\}$ wins a session iff the session finishes in F_C. A strategy of a player is a subset of the player's possible moves. A winning strategy for a player is a strategy of the player which always leads to the player's win: the player wins every session in which he/she implements this strategy instead of all moves.

Proposition 1.
1. No CTL formula can express existence of winning strategies in finite games.
2. μC formula $\mu\ x.\ (P_A \vee \langle M_A \rangle x \vee (\langle M_B \rangle true \wedge [M_B]x))$ expresses existence of winning strategies for player A against B in games with terminal positions.

Proof. We would like to prove the first part only. We can consider CTL formulae without constructions **AG**, **AF**, **EG**, and **EF** at all due to the following equivalences:

$$\begin{array}{ll} \mathbf{AF}\phi \leftrightarrow \mathbf{A}(true\mathbf{U}\phi) & \mathbf{EG}\phi \leftrightarrow \neg\mathbf{AF}(\neg\phi) \\ \mathbf{EF}\phi \leftrightarrow \mathbf{E}(true\mathbf{U}\phi) & \mathbf{AG}\phi \leftrightarrow \neg\mathbf{EF}(\neg\phi) \end{array}$$

Let us define nesting for CTL formulae of this kind as follows:
$nest(true) = nest(false) = 0$,
$nest(r) = 0$ for every propositional variable r,
$nest(\neg\phi) = nest(\phi)$,
$nest(\phi \wedge \psi) = nest(\phi \vee \psi) = \max\{nest(\phi), nest(\psi)\}$,
$nest(\mathbf{A}(\phi\mathbf{U}\psi)) = nest(\mathbf{E}(\phi\mathbf{U}\psi)) = \max\{nest(\phi), nest(\psi)\}$,
$nest(\mathbf{AX}\phi) = nest(\mathbf{EX}\phi) = 1 + nest(\phi)$.
Then let us consider a game NEG (Fig. 1), where A/B-line represents positions where a player A/B has moves, downward/upward arrows represent moves of A/B, W_A/W_B represents a single winning position for A/B. Then $(-k_A) \models_{NEG} \phi \Leftrightarrow (-l_B) \models_{NEG} \phi$ holds for every CTL formula ϕ and for all $k, l > nest(\phi)$.

$$\begin{array}{lcc}
A: \ldots (-i-1) & \overbrace{(-i) \ldots (-1) \quad \overset{W_A}{(0)}}^{NEG_i} \\
\ldots \qquad\quad \times & \times \ldots \quad \times \\
B: \ldots (-i-1) & \underbrace{(-i) \ldots (-1) \quad \overset{W_B}{(0)}}_{NEG_i}
\end{array}$$

Fig. 1. Model NEG and NEG_i

Hence for every formula of CTL there exists a non-positive number prior to which the formula is a boolean constant in the NEG. Let us also remark that no CTL formula can distinguish positions of a finite game NEG_i (fig. 1) from correspondent positions of NEG. But A has a winning strategy in all even integers on A-line and in all odd integers on B-line. Hence no CTL formula can express an existence of a winning strategy for a player A in positions of finite games NEG_i for $i \geq 0$. Thus the proof of proposition 1 is **over.**

Another evidence of μC expressive power comes from comparison with the Second-order logic of several monadic Successors of M. Rabin (**S(n)S**-Logic) [12, 13, 2]: both logics enjoy equal expressive power on infinite trees [14]. But in spite of this, μC fails to express some very natural properties discussed below. For resolving the problem with a deficit of expressive power of μC , the second order propositional program logic 2M of C. Stirling [17] uses second order quantifiers $\forall/\exists$ over propositional variables and reachability modalities $\Box/\Diamond$ upon strongly connected components of models.

Proposition 2.
1. No μC formula can express commutativeness of composition of action symbols in finite models.
2. 2M formula $\forall p.\big(([a][b]p) \leftrightarrow ([b][a]p)\big)$ expresses commutativeness of composition of action symbols a and b in models.

But some other very natural and useful properties are still inexpressible in 2M. For instance 2M can not express the weakest precondition for inverse actions as it is defined below. Assume that a propositional variable p is interpreted in a model by a set of states P, and an action symbol r is interpreted by a binary relation R on states. The weakest pre-condition for inverse of r with respect to a post-condition p in this model is the following state of states $\{t : \forall\ s.((s,t) \in R \Rightarrow s \in P)\}$. We would like to suggest a another Second-Order Elementary Propositional Dynamic Logic (SOEPDL) for handling this phenomenon. A single difference between 2M and SOEPDL is interpretation of reachability modalities $\Box/\Diamond$: in SOEPDL they means "for every/some state in the model".

Proposition 3.
1. No 2M formula can express in finite models the weakest pre-conditions for inverse of action symbol with respect to propositional variable.
2. SOEPDL formula $\exists q.\big(\Box(\langle a\rangle q \to p) \wedge q\big)$ expresses the weakest pre-conditions for inverse of action symbol a with respect to propositional variable p in models.

2 Expressiveness vs. Model Checking

Theorem 4.
CTL $<$ μC $<$ 2M $<$ SOEPDL where all expressibilities have linear complexity and all inexpressibilities can be justified in finite models.

Proof.
First, CTL $<\mu$C since it is well known that CTL $\leq \mu$C and, in accordance with proposition 1, there is μC formula, which is not equivalent to any CTL formula in finite models.

Next, μC $<$ 2M since $(\mu p.\phi) \leftrightarrow (\forall p.(\Box(\phi \to p) \to p))$, $(\nu p.\phi) \leftrightarrow (\exists p.(\Box(p \to \phi) \wedge p))$, and, in accordance with proposition 2, there is 2M formula, which is not equivalent to any μC formula in finite models.

Finally, 2M $<$ SOEPDL since reachability modalities can be expressed in terms of PDL programs as $\Box\phi \leftrightarrow [(\cup_{a\in Act}a)*]\phi$ and $\Diamond\phi \leftrightarrow \langle(\cup_{a\in Act}a)*\rangle\phi$, and, in accordance with proposition 3, there is SOEPDL formula, which is not equivalent to any 2M formula in finite models.

Thus the proof of the theorem 4 is **over.**

An "internal" characteristic of the expressive power of SOEPDL in terms of other propositional program logics correlates with an "external" one: we demonstrate that SOEPDL is as expressive as Second Order Logic of **n** Monadic Successors of M. Rabin (**S(n)S**-Logic) [12, 13, 2]. We would not like to give a complete definition of **S(n)S**-Logic, but would like to point out that we use action symbols as symbols of monadic functions (i.e., "successors") and exploit functional models, where action symbols are interpreted as total monadic functions. In these settings functional models are just special kind of SOEPDL models. Boolean values of formulae of **S(n)S**-Logic in functional models depend on values of free individual variables only. Thus boolean values of formulae of **S(n)S**-Logic with a single (at most) free individual variable depend on values of this single variable. In this setting semantics of formulae of **S(n)S**-Logic with a single (at most) free first order variable is a state-based semantics as well as semantics of SOEPDL and it is possible to compare semantics of formulae of **S(n)S**-Logic of this kind and SOEPDL formulae in terms of equivalence.

Theorem 5. *Expressive powers of SOEPDL and* **S(n)S**-*Logic are linear time equivalent in the following sense:*

- *for every formula of* **S(n)S**-*Logic with a single (at most) free first-order variable it is possible construct in linear time an equivalent in all functional models formula of SOEPDL;*
- *for every formula of SOEPDL it is possible construct in linear time an equivalent in all functional models formula of* **S(n)S**-*Logic with a single (at most) free first-order variable.*

Expressiveness is a particular dimension where we can compare a power of program logics. Another possible dimension is model checking power. Let us discuss it below. Global (model) checking consists in a calculation of the set

– $(s, S_x, ...S_z, (\psi_1 \overset{\wedge}{\vee} \psi_2)) \to (s, S_x, ...S_z, \psi_i)$ for every $i \in \{1, 2\}$;
– $(s, S_x, ...S_z, (\frac{[a]}{\langle a \rangle} \psi)) \to (t, S_x, ...S_z, \psi)$ for every t such that $(s, t) \in I(a)$;
– $(s, S_x, ...S_z, (\frac{\Box}{\Diamond} \psi)) \to (t, S_x, ...S_z, \psi)$ for every t in M;
– $(s, S_x, ...S_y, ...S_z, (\frac{\forall}{\exists} y.\psi)) \to (s, S_x, ...S, ...S_z, \psi)$ for every $S \subseteq D$.

Fig. 2. Moves of Falsifier/Verifier

of all states of an input model where an input formula is valid. Local (model) checking is checking an input formula in an input state of an input model. If LG is a logic and MD is a class of models, then a model checker for LG×MD is a program (algorithm) which can check all LG formulae in all MD models. Assume LG′ is a propositional program logic and MC′ be a model checker for LG′×MD. Assume we would like to check formulae of another propositional program logic LG″ in MD models. A first move is to try to reuse MC′, i.e., to force MC′ to do this job instead of expensive and risky design, implementation and validation of a new model checker MC″ for LG″×MD. If LG″ $\leq$ LG′ then the work is done. The question is: when LG″ $\not\leq$ LG′, is it still possible to reuse MC′ for LG″×MD? In particular, whether a model checker of CTL formulae in finite models can be utilized for solving board games in spite of lack of expressive power of CTL?

Let ξ be SOEPDL formula. Without loss of generality we can assume that variables bounded by different quantifiers are different (so there are no name collisions). Moreover we can assume that negations in the formula are applied to propositional variables only since the following equivalences hold:
$(\neg(\neg\phi)) \leftrightarrow \phi$
$(\neg(\phi \wedge \psi)) \leftrightarrow ((\neg\phi) \vee (\neg\psi))$ $(\neg(\langle a \rangle \phi)) \leftrightarrow ([a](\neg\phi))$
$(\neg(\Diamond\phi)) \leftrightarrow (\Box(\neg\phi))$ $(\neg(\exists p.\phi)) \leftrightarrow (\forall p.(\neg\phi))$
Let x, ... z be a list of all bounded propositional variables in ϕ. A subformula of ξ is a formula, which is (syntactically) a part of ϕ. A subformula is said to be *conjunctive* iff it has one of the following forms: $\phi \wedge \psi$, $[a]\phi$, $\Box\phi$ or $\forall p.\phi$. A subformula is said to be *disjunctive* iff it has one of the following forms: $\phi \vee \psi$, $\langle a \rangle \phi$, $\Diamond\phi$ or $\exists p.\phi$. Propositional variables and their negations (and only they) are neither conjunctive nor disjunctive subformulae. Let M be a finite model with a domain D and an interpretation I.

We are going to construct a Hintica-like finite game $G(M, \xi)$ of two players *Falsifier* and *Verifier*. Positions in this game $G(M, \xi)$ are tuples $(s, S_x, ...S_z, \psi)$ where $s \in D$ is a current state, $S_x, ...S_z \subseteq D$ are current interpretations for x, ... y, and ψ is a verified subformula of ξ. Falsifier/Verifier has moves of 4 kinds (Fig. 2) related conjunctive/disjunctive subformulae respectively, and wins in positions of 6 kinds (Fig. 3). Intuitively: Verifier is trying to validate a formula in a state while a rival Falsifier is trying to falsify the formula in a state.

The following is easy to prove by induction on formula structure.

Proposition 6. *For every finite model $M = (D, I)$ and every SOEPDL formula ξ there exists a finite game $G(M, \xi)$ of two players Falsifier and Verifier such that the following holds for every state $s \in D$ and every subformula ϕ of ξ:*

– $(s, S_x, ...S_z, \frac{false}{true})$,
– $(s, S_x, ...S_z, p)$ where p is a free propositional variable and $s \frac{\notin}{\in} I(p)$,
– $(s, S_x, ...S_z, \neg p)$ where p is a free propositional variable and $s \frac{\in}{\notin} I(p)$,
– $(s, S_x, ...S_z, (\frac{\langle a \rangle}{[a]}\psi))$ iff $(s, t) \notin I(a)$ for every t;
– $(s, S_x, ...S_y, ...S_z, y)$, where $s \frac{\notin}{\in} S_y$,
– $(s, S_x, ...S_y, ...S_z, \neg y)$, where $s \frac{\in}{\notin} S_y$.

Fig. 3. Winning positions of Falsifier/Verifier

Verifier has a winning strategy against Falsifier in a position $(s, I(x), ...I(z), \phi)$ iff $s \models_M \phi$ (where $x, ...z$ is a list of all bounded variables of ξ). The game can be constructed in time $O(2^{d \times f} \times d \times f)$, where d is amount of states and f is formula size.

It immediately implies the following sufficient
Criterion *Let L' be a program logic and MC be a model checker for this logic L' in finite models which can check an existence of a winning strategy for a player against a counterpart in positions of finite games with time complexity $T(m)$, where m is an overall game size. Let L'' be another logic which is expressively equivalent to a fragment of SOEPDL and $C(f)$ and $S(f)$ be time and space complexity of a translation L'' formulae into SOEPDL formulae, where f is a formula size. Then MC can be reused for checking L'' formulae in finite models and upper time bound for model checking L'' in finite models is $max\{C(f),\ T(O(2^{d \times S(f)} \times d \times S(f)))\}$, where d is amount of states in the model.*

A consequence of the above criterion, proposition 1 and theorem 4 is

Theorem 7. *Let MC be a model checker which implements linear in an overall model size model checking algorithm for the following formula $\mu\ x.\ (p \vee \langle a \rangle x \vee (\langle b \rangle true \wedge [b]x))$ of μC in finite models. MC can be reused for checking all formulae of SOEPDL, 2M, and μC in finite models with upper time bound $exp(d \times f)$, where d is amount of states in a model and f is formula size.*

We would like to remark that a formula $\mu\ x.\ (p \vee \langle a \rangle x \vee (\langle b \rangle true \wedge [b]x))$ from the theorem above can be checked in finite models in linear time [6].

We would like to generalize the above theorem 7 as follows below. A class of models MD is closed with respect to Cartesian products iff for all models M' and M'' in MD, every model M with $D_M = D_{M'} \times D_{M''}$ is in MD also. A class of models MD is *closed with respect to power-sets* iff for every model M' in MD, every model M with $D_M = \mathcal{P}(D_{M'})$ is in MD also. Due to space limitations we would like to present the following theorem without proof.

Theorem 8. *Let MD be a class of models which contains all finite models and is closed with respect to Cartesian products and power-sets. Let MC be a model checker which can check (at least) the following formula ($\mathbf{EF}p$) of CTL in models in MD. Then MC can be reused for checking all formulae of SOEPDL, 2M, μC and CTL in models in MD.*

3 Conclusion

We began this research from a particular problem whether a model checker for CTL formulae in finite models can be used for solving board games. Now we content our curiosity: in principle yes it is (theorem 8), in spite of lack of expressive power (proposition 1). Theoretical method suggested for it is very simple from algebraic viewpoint, since it exploits Cartesian products and power set operation. But it is too expensive in computation complexity and impractical due to double exponential blow up of a model (power set operation is used twice).

In general, in spite of algorithmical inefficiency of presented results, contribution of this paper is two-folds. First we study expressive and model checking power of the classical propositional logic or a basic propositional modal logic **K** [3, 15], extended by transitive closure (CTL), fixpoints (μC), and second-order monadic quantification (2M and SOEPDL). We consider this study as a propositional counterpart of a popular research topic in descriptive complexity, where expressive power and finite spectra are examining for First-Order Logic, extended by transitive closure (FOL+TC), fixpoints (FOL+FP), and second-order quantification (SOL) [9].

Next contribution consists in model checkers reuse. Let us discuss it with some details in the next paragraph. Assume we have a reliable model checker MC for CTL in finite models, but we would like to check in finite models formulae of more powerful combined logic (CTL+PDL) extended by program intersection $\cap$ (it is essential for representation and reasoning about *distributed* knowledge in logic of knowledge [8]). This extended logic can not be expressed neither in CTL nor in μC (due to presence of program intersection), but in SOEPDL only as follows: $\forall p.([\alpha \cap \beta]p \leftrightarrow ([\alpha]p \wedge [\beta]p))$. Does it imply a necessity to implement a new model checker in stead of MC? Or does it imply that code revision and patching of MC are inevitable? - Not at all, as follows from equal model checking power of CTL and SOEPDL! One can try to encode extensions in models instead of risky implementation and patching. If it can be done in feasible time and space in terms of MC's input language for models, then just implement it as a preprocessor for MC, and reuse MC for extended logic. In particular, this approach is valid for (CTL+PDL) extended by program intersection $\cap$, since an encoding in this particular case is linear in time and space. But in general, better analysis when model checkers for CTL and μC in finite models can be reused for model checking other logics in finite models without loss of efficiency is a research topic for a perspective.

Finally we would like to remark that close connections between model checking for μC and special games have been extensively examined [16, 17, 18]. In particular, [16] has defined infinite model checking games and established an equivalence of local model checking to an existence of winning strategies. Then [17] has defined *finite* fixed point games and characterized indistinguishability of processes by means of formulae with bounded amounts of modalities and fixpoints in terms of winning strategies with bounded amounts of moves. Paper [18] has exploited model-checking games for practical efficient local model checking. Paper [17] has defined also logic 2M, corresponding games and established indistinguishability of states by means of formulae with bounded amounts of modalities and quantifiers in terms of winning strategies with bounded amounts of moves. So it is possible to summarize that [16, 17, 18] have exploited infinite games for local model checking μC in infinite models. In contrast, we exploit games with terminal positions (basically, finite games) for forcing model checkers for CTL to check more expressive logics μC , 2M, and SOEPDL.

References

1. Baldamus M, Schneider K., Wenz M., Ziller R. *Can American Checkers be Solved by Means of Symbolic Model Checking?* Electronic Notes in Theoretical Computer Science, v.43, `http://www.elsevier.nl/gej-ng/31/29/23/show/Products/notes/`
2. Börger E., Grädel E., Gurevich Y. *The Classical Decision Problem.* Springer, 1997.
3. Bull R.A., Segerberg K. *Basic Modal Logic.* Handbook of Philosophical Logic, v.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, v.98, n.2, 1992, p.142-170.
5. Clarke E.M., Grumberg O., Peled D. Model Checking. MIT Press, 1999.
6. Cleaveland R., Klain M., Steffen B. *Faster Model-Checking for Mu-Calculus.* Lecture Notes in Computer Science, v.663, 1993, p.410-422.
7. Emerson E.A. *Temporal and Modal Logic. Handbook of Theoretical Computer Science*, v.B, Elsevier and The MIT Press, 1990, 995-1072.
8. Fagin R., Halpern J.Y., Moses Y., Vardi M.Y. Reasoning about Knowledge. MIT Press, 1995.
9. Immerman N *Descriptive Complexity: a Logician's Approach to Computation.* Notices of the American Mathematical Society, v.42, n.10, p.1127-1133.
10. Kozen D. *Results on the Propositional Mu-Calculus.* Theoretical Computer Science, v.27, n.3, 1983, p.333-354.
11. Kozen D., Tiuryn J. *Logics of Programs.* Handbook of Theoretical Computer Science, v.B, Elsevier and The MIT Press, 1990, 789-840.
12. Rabin M.O. *Decidability of second order theories and automata on infinite trees.* Trans. Amer. Math. Soc., v.141, 1969, p.1-35.
13. Rabin M.O. *Decidable Theories.* in Handbook of Mathematical Logic, ed. Barwise J. and Keisler H.J., North-Holland Pub. Co., 1977, 595-630.
14. Schlingloff H. *On expressive power of Modal Logic on Trees.* LNCS, v.620, 1992, p.441-450.
15. Stirling C. *Modal and Temporal Logics.* Handbook of Logic in Computer Science, v.2, Claredon Press, 1992, p.477-563.
16. Stirling C. *Local Model Checking Games.* Lecture Notes in Computer Science, v.962, 1995, p.1-11.
17. Stirling C. *Games and Modal Mu-Calculus.* Lecture Notes in Computer Science, v.1055, 1996, p.298-312.
18. Steven P., Stirling C. *Practical Model Checking Using Games.* Lecture Notes in Computer Science, v.1384, 1998, p.85-101.

This article was processed using the LaTeX macro package with LLNCS style