Formal Language Constrained Reachability and Model Checking Propositional Dynamic Logics*

Roland Axelsson[1] and Martin Lange[2]

[1] Dept. of Computer Science, University of Munich, Germany
[2] School of Electr. Eng. and Computer Science, University of Kassel, Germany

Abstract. We show interreducibility under (Turing) reductions of low polynomial degree between three families of problems parametrised by classes of formal languages: the problem of reachability in a directed graph constrained by a formal language, the problem of deciding whether or not the intersection of a language of some class with a regular language is empty, and the model checking problem for Propositional Dynamic Logic over some class of formal languages. This allows several decidability and complexity results to be transferred, mainly from the area of formal languages to the areas of modal logics and formal language constrained reachability.

1 Introduction

This paper investigates three families of decision problems from the domains of formal language theory, digraph reachability, and model checking. Each family is parametrised by a class $\mathcal{L}$ of formal languages which can be any class but we are mainly concerned with known and natural classes like the regular, context-free, context-sensitive languages, and also some equally natural but lesser known classes. We will always assume that there is a finite representation of any member of that class, for instance a finite-state automaton for a regular language or a context-free grammar for a context-free language, etc. The three families of problems are the following.

i) REG-Intersection for $\mathcal{L}$: determine for a given language $L \in \mathcal{L}$ and a regular language R, whether or not $L \cap R$ is empty.
ii) $\mathcal{L}$-Reachability: decide for a given directed graph with edge labels and node predicates whether or not there is a path from a designated source node to a designated target area s.t. the path is described by a given language $L \in \mathcal{L}$.
iii) Model checking PDL[$\mathcal{L}$]: decide for a given state of a Kripke structure and a given formula of Propositional Dynamic Logic over $\mathcal{L}$ whether or not the state satisfies the formula.

* The European Research Council has provided financial support under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no 259267.

G. Delzanno and I. Potapov (Eds.): RP 2011, LNCS 6945, pp. 45–57, 2011.

These problems have been considered each on its own so far, and the state of the art in knowing about decidability and complexity of these problems is the following.

(i) Closure under intersection with a regular language and decidability of the emptiness problem are two of the most important features usually investigated with any class of formal languages. Note that a class that is closed under intersections with regular languages and has a decidable emptiness problem also has a decidable REG-intersection problem. The converse need not necessarily be the case but to the best of our knowledge there is no natural class which would witness this. It can safely be said that the REG-intersection problem is well-understood in the domain of formal languages.

(ii) Digraph reachability is of course one of the most fundamental problems in computer science and related disciplines. The use of constraints restricting the paths under which certain vertices should be reachable has been outlined for multi-modal path planning for instance [5]. Complexity and decidability issues have been investigated with respect to a class of formal languages used to constrain the paths. It has been found that using context-free languages as opposed to no languages or regular ones increases the complexity from NLOGSPACE to PTIME, and using context-sensitive languages, this problem becomes undecidable [5]. To the best of our knowledge, the space between context-free and context-sensitive languages has not been looked at from the perspective of formal language constrained reachability problems.

(iii) Propositional Dynamic Logic (PDL) has been introduced by Fischer and Ladner [8] as a modal logic for reasoning about programs. Various applications of PDL have been identified, for instance in program verification because of its similarity to the branching-time temporal logic CTL and its apparent relation to Hoare logic; in knowledge representation and artificial intelligence because the test-free fragment for example turns out to equal the description logic $\mathcal{ALC}_{\text{reg}}$ [10], because it can be used to reason about knowledge [6] or about actions [25], etc. While much attention is being paid to its satisfiability problem, its model checking problem also has applications. It mainly occurs in automatic program verification, and certain inference problems in description logics for example can be reduced to model checking problems and then be tackled using database technology [3].

The original PDL is in fact *PDL over regular programs*—here written as PDL[REG]—since the programs which are interpreted as binary relations on program states, are built from atomic ones using the constructors union, composition and iteration. Such programs are denoted syntactically using regular expressions, and it is imminent that other formalisms for describing formal languages can be used instead, too.

Variants of PDL over richer classes of formal languages have been studied with a focus on their satisfiability problems [12,13,14,11]. This is undecidable for PDL[CFL] – *PDL over context-free languages* – already [12]. On the other hand, model checking for PDL[CFL] is PTIME-complete [18] as it is for PDL[REG]

[8], and larger classes as parameters have not been considered yet under this aspect.

In this paper we show that these problems are (Turing-)interreducible to each other in polynomial time: one reduction is a genuine Turing reduction of quadratic time, the others are many-one reductions of linear time. The constructions are simple, and so are their proofs of correctness. However, these simple constructions pave the way for a number of new decidability and complexity results on formal language constrained reachability analysis as well as on model checking extensions of Propositional Dynamic Logic. As a consequence, the gap between decidability and undecidability in terms of the class of formal languages used as the parameter has been narrowed down significantly: with the results obtained here we now know that it lies between a large subclass of CSL known as the multi-stack visibly pushdown languages (MVPL) and CSL itself.

The paper is organised as follows. Sect. 2 introduces the three problems formally. Sect. 3 motivates the study of the interconnection between these three problems and their respective areas by presenting exemplary applications of these problems. It gives a brief insight into the fact that these three problems, despite being very much related as decision problems, have been studied independently in different domains. Sect. 4 proves the interreducibility. New decidability and complexity results about model checking and reachability problems are derived in Sect. 5 from corresponding language-theoretic problems using this interreducibility. Finally, Sect. 6 provides a summary of the complexity and decidability results known in these areas.

2 Preliminaries

Classes of formal languages and their representations. Let Σ be a finite alphabet. As usual, a formal language is a $L \subseteq \Sigma^*$, and a class of formal languages is a $\mathcal{L} \subseteq 2^{\Sigma^*}$. We do not want to advertise nor restrict the use of a particular specification formalism for formal languages like automata, grammars, algebraic expressions, systems of equations, etc. We therefore identify a class $\mathcal{L}$ of formal languages with a class of its *acceptors* and restrict our attention to classes which can be represented by such acceptors. This means that we can assume a size measure $\|L\|$ which is a finite value for any L, even though it may contain infinitely many words. For instance, for $\mathcal{L} = \mathrm{REG}$ this may be the size of a smallest nondeterministic finite automaton recognising L.

We make another very reasonable assumption on each $\mathcal{L}$: given an $L \in \mathcal{L}$, its alphabet must be computable in time $\mathcal{O}(\|L\|)$.[1] We write $\Sigma(L)$ to denote the alphabet that is underlying L.

The REG-intersection problem for classes of formal languages. Remember that the *non-emptiness problem* for a class $\mathcal{L}$ of languages is the following: given a

[1] This is true of virtually all known specification formalisms for formal languages and only precludes strange acceptors like encrypted strings representing automata, etc.

suitably represented $L \in \mathcal{L}$, decide whether or not $L \neq \emptyset$. Furthermore, a class $\mathcal{L}$ is *closed under intersections with regular languages* if for every $L \in \mathcal{L}$ and every regular language R we have $L \cap R \in \mathcal{L}$. These are combined into a decision problem which is of particular interest here. We assume familiarity with the theory of regular languages. Note that $L(\mathcal{A})$ denotes the language recognised by the automaton $\mathcal{A}$.

Definition 1. The problem of *non-emptiness of intersection with a regular language – REG-intersection problem* for short – for $\mathcal{L}$ is the following: given a suitably represented $L \in \mathcal{L}$ and a non-deterministic finite automaton (NFA) $\mathcal{A}$ over Σ, decide whether or not $L \cap L(\mathcal{A}) \neq \emptyset$.

Clearly, if a class of languages is closed under intersections with regular languages and has a decidable non-emptiness problem, then its REG-intersection problem is decidable, too. The converse may not be true in general.[2] Furthermore, if a class of languages is closed under intersections with regular languages but has an undecidable non-emptiness problem (like CSL for instance) then its REG-intersection problem is necessarily also undecidable.

Kripke structures, labeled digraphs, and words. Let Σ be a finite set of symbols and $\mathcal{P}$ be a countably infinite set of propositional constants. A *Kripke structure* is a triple $\mathcal{T} = (\mathcal{S}, \to, \ell)$, where $\mathcal{S}$ is a set of states, $\to \subseteq \mathcal{S} \times \Sigma \times \mathcal{S}$ is a transition relation and $\ell : \mathcal{S} \to 2^{\mathcal{P}}$ labels each state with a set of propositions that are true in that state. We write $s \xrightarrow{a} t$ instead of $(s, a, t) \in \to$. We will restrict ourselves to finite Kripke structures, i.e. those for which $|\mathcal{S}|$ is finite.

The accessibility relation $\to$ is inductively extended to words over Σ as follows.

$$\begin{aligned} s \xrightarrow{\epsilon} t \quad &\text{iff} \quad s = t \\ s \xrightarrow{aw} t \quad &\text{iff} \quad \exists u \in \mathcal{S} \text{ with } s \xrightarrow{a} u \text{ and } u \xrightarrow{w} t \end{aligned}$$

An *edge-labeled directed graph* is a Kripke structure $\mathcal{T}$ as above such that $\ell(s) = \emptyset$ for all $s \in \mathcal{S}$. In the following, when we speak of a graph it is implicitly to be understood as an edge-labeled directed graph. We will denote such a structure like a Kripke structure but leaving out the labeling function, i.e. as $\mathcal{T} = (\mathcal{S}, \to)$.

The $\mathcal{L}$-reachability problem for a class of formal languages $\mathcal{L}$.

Definition 2. Let $\mathcal{L}$ be a class of languages over Σ. The *$\mathcal{L}$-reachability problem* is the following: given a graph $\mathcal{T} = (\mathcal{S}, \to)$, a state $s \in \mathcal{S}$, a set of target states $T \subseteq \mathcal{S}$ and a suitably represented $L \in \mathcal{L}$, decide whether or not there is a $w \in L$ and a $t \in T$ s.t. $s \xrightarrow{w} t$.

We also say that T is L-reachable from s in $\mathcal{T}$ if these form a positive instance of the $\mathcal{L}$-reachability problem.

[2] However, we are unaware of any (necessarily strange) class of languages that witnesses the failure of the converse direction. It also does not matter for our purposes here.

Propositional Dynamic Logic over a class of formal languages. Formulas of Propositional Dynamic Logic (with tests) over a class $\mathcal{L}$ of formal languages over some finite alphabet Σ — PDL[$\mathcal{L}$] — are defined recursively as the least set Form satisfying the following.

(i) $\mathcal{P} \subseteq \mathsf{Form}$
(ii) If $\varphi \in \mathsf{Form}$ and $\psi \in \mathsf{Form}$ then $\neg\varphi \in \mathsf{Form}$ and $\varphi \vee \psi \in \mathsf{Form}$.
(iii) If L is a language over the alphabet $\Sigma \cup \{\psi? \mid \psi \in \mathsf{Form}\}$ s.t. $|\Sigma(L)| < \infty$ and $\varphi \in \mathsf{Form}$ then $\langle L\rangle\varphi \in \mathsf{Form}$.

We use the usual abbreviations: $\mathtt{tt} := q \vee \neg q$ for some $q \in \mathcal{P}$, $\mathtt{ff} := \neg\mathtt{tt}$, $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$, $\varphi \rightarrow \psi := \neg\varphi \vee \psi$, $[L]\varphi := \neg\langle L\rangle\neg\varphi$. We define $|\varphi|$ as the number of different subformulas of φ plus the sum over all $\|L\|$ s.t. $\langle L\rangle\psi$ is a subformula of φ for some ψ.

Suppose L is a language over $\Sigma \cup \{\psi? \mid \psi \in \Phi\}$ for some finite $\Phi \subset \mathsf{Form}$. We then write $\mathit{Tests}(L)$ for this set Φ. By the assumption made above about L being represented reasonably, we can also assume that $\mathit{Tests}(L)$ can be computed in time $\mathcal{O}(\|L\|)$.

Formulas of PDL[$\mathcal{L}$] are interpreted in states s of Σ-labeled Kripke structures $\mathcal{T} = (\mathcal{S}, \rightarrow, \ell)$ as follows.

$$
\begin{array}{rll}
\mathcal{T}, s \models q & \text{iff} & q \in \ell(s) \\
\mathcal{T}, s \models \neg\varphi & \text{iff} & \mathcal{T}, s \not\models \varphi \\
\mathcal{T}, s \models \varphi \vee \psi & \text{iff} & \mathcal{T}, s \models \varphi \text{ or } \mathcal{T}, s \models \psi \\
\mathcal{T}, s \models \langle L\rangle\varphi & \text{iff} & \text{there are } w \in L \text{ and } t \in \mathcal{S} \text{ s.t. } s \xrightarrow{w}{}' t \text{ and } \mathcal{T}, t \models \varphi \text{ where} \\
 & & \quad \rightarrow' := \rightarrow \cup \{(u, \psi?, u) \mid u \in \mathcal{S}, \psi \in \mathit{Tests}(L), \text{ and } \mathcal{T}, u \models \psi\}
\end{array}
$$

Definition 3. The *model checking problem* for PDL[$\mathcal{L}$] is the following: given a Kripke structure $\mathcal{T} = (\mathcal{S}, \rightarrow, \ell)$, a state $s \in \mathcal{S}$ and a formula $\varphi \in$ PDL[$\mathcal{L}$], decide whether or not $\mathcal{T}, s \models \varphi$ holds.

3 Applications

We will briefly give some examples of the use of the three problems introduced in the previous section showing that each of them has attracted interest independent of the others.

Verification of Programs with Stack Inspection. In order to detect access violations in safety critical routines, inspection of the call stack may become necessary, e.g. in case of nested calls, where the initial call came from a method without the required permission. This has been implemented for instance in the runtime access control mechanism of JDK 1.2. In [22], such programs are modeled as the set of possible sequences of the call stack w.r.t. the program flow, called traces. The set of possible traces L_{tr} is an indexed language. The class IL of indexed languages [1] forms a subclass of the context-sensitive languages which properly

includes the context-free languages and possesses some nice closure properties and decidability results.

One considers a regular language L_{safe} representing the set of safe traces. The verification itself is then performed by checking $L_{\mathsf{tr}} \subseteq L_{\mathsf{safe}}$, i.e. $L_{\mathsf{tr}} \cap \overline{L_{\mathsf{safe}}} = \emptyset$. and therefore is an instance of the REG-intersection problem for IL. Note that REG is closed under complement.

CFL- and REG-Reachability. The REG-reachability problem is at the core of several applications in network routing and intermodal route planning for instance [5]. It is known that the reachability problem in directed graphs when constrained with a regular language is not more difficult than the plain digraph reachability problem, i.e. NLOGSPACE-complete. However, it becomes PTIME-complete when the constraints are formed by context-free languages [5]. Such reachability problems, in particular for context-free languages have important applications in static analysis [26]. CFL-reachability for instance is used in type-based polymorphic flow analysis [7], field-sensitive points-to analysis or interprocedural dataflow analysis [27].

It is worth investigating decidability and complexity issues for classes beyond CFL which may allow more refined program analyses. Only little is known in this area so far, namely it is known that CSL-reachability is undecidable [5] which is very easily seen to be the case.

Model Checking PDL[CFL] in Abstract Interpretation. Consider the following system of mutually recursive functions where "+" denotes nondeterministic choice, ";" denotes sequential composition, and "term" denotes an anonymous terminating function.

$$\begin{aligned}
\mathsf{f}_0 &:= \mathsf{f}_2;\mathsf{f}_3 + \mathsf{f}_2;\mathsf{f}_1\\
\mathsf{f}_1 &:= \mathsf{f}_3;\mathsf{f}_1 + \mathsf{f}_2;\mathsf{f}_3 + \mathsf{f}_1;\mathsf{f}_3\\
\mathsf{f}_2 &:= \mathsf{f}_1;\mathsf{f}_2 + \mathsf{f}_2;\mathsf{f}_3 + \mathsf{term}\\
\mathsf{f}_3 &:= \mathsf{f}_1;\mathsf{f}_1 + \mathsf{term}
\end{aligned}$$

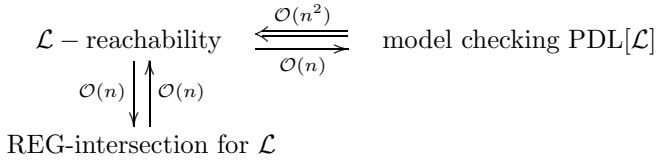
The function f_0 is the entry point of the system. Suppose we were interested in detecting whether on all possible system executions the call of f_3 is preceded by a successful return of f_1 (security check). Note that the stack behaviour, i.e. the sequences of function calls and returns is non-regular in general (for a non-fixed number of functions). We state the property we wish to verify as the regular expression $L_{\mathsf{safe}} = \Sigma^* c_1 \Sigma^* r_1 \Sigma^* c_3 \Sigma^*$, where a call of function $\mathsf{f_i}$ is indicated by c_i, a return by r_i respectively. It is possible to use abstract interpretation and overapproximate the system of recursive function into a one-state transition system with looping transitions for all elements in Σ. In order to restrict this overapproximation to non-spurious runs one can consider the context-free grammar

$$\begin{aligned}
F_0 &\to c_0 F_2 F_3 r_0 \mid c_0 F_2 F_1 r_0\\
F_1 &\to c_1 F_3 F_1 r_1 \mid c_1 F_2 F_3 r_1 \mid c_1 F_1 F_3 r_1\\
F_2 &\to c_2 F_1 F_2 r_2 \mid c_2 F_2 F_3 r_2 \mid c_2 r_2\\
F_3 &\to c_3 F_1 F_1 r_3 \mid c_3 r_3
\end{aligned}$$

which is straight-forwardly derived from the recursive functions. Safety of the system is then established by checking the PDL[CFL] property $\varphi_{\mathsf{safe}} = \neg\langle L(G) \cap \overline{L_{\mathsf{safe}}}\rangle\mathtt{tt}$. It is easy to see that the only state s does not satisfy φ_{safe}: $F_0 \Rightarrow c_0F_2F_1r_0 \Rightarrow^3 c_0c_2c_2r_2c_3r_3r_2F_1r_0$. Every derivation continuing from this point will end in a violation of L_{safe}, because every derivation from F_1 will be prefixed by c_1.

4 The Connection between the Three Problems

We first show that the three problems defined in Sect. 2 are interreducible onto each other. This is done as follows.



A single line from X to Y denotes a many-one reduction from X into Y transfering lower bounds along the arrow and upper bounds in the opposite direction. A double line denotes a Turing reduction transferring only an upper bound down the arrow but not a lower bound up the arrow.

We will begin with the forth and back between $\mathcal{L}$-reachability and model checking PDL[$\mathcal{L}$] (Lemmas 1 and 2), and then show the linear-time equivalence of $\mathcal{L}$-reachability and REG-intersection for $\mathcal{L}$ (Lemmas 3 and 4). Note that a circular series of reductions would not save any effort since the reduction from REG-intersection to model checking is very easily obtained as the composition of the two arrows via reachability. Moreover, a reduction from model checking to REG-intersection would also only be a Turing reduction, and it is not conceptually simpler than the composition of the two respective arrows via reachability.

4.1 Forth and Back between Graphs and Formulas

Lemma 1. *Let $\mathcal{L}$ be a class of languages. The $\mathcal{L}$-reachability problem reduces in linear time to the model checking problem for PDL[$\mathcal{L}$].*

Proof. Let $\mathcal{T} = (\mathcal{S}, \to)$ be a graph, $s \in \mathcal{S}$ and $T \subseteq \mathcal{S}$. Now take a proposition q_T and let $\mathcal{P} := \{q_T\}$. Define $\mathcal{T}' = (\mathcal{S}, \to, \ell')$ s.t. for all $u \in \mathcal{S}$: $q_T \in \ell'(u)$ iff $u \in T$. It is not hard to see that, for any $L \in \mathcal{L}$, there is a $w \in L$ and a $t \in T$ with $s \xrightarrow{w} t$ iff $\mathcal{T}', s \models \langle L\rangle q_T$. Furthermore, both $\mathcal{T}'$ and $\langle L\rangle q_T$ can be constructed in time $\mathcal{O}(|\mathcal{T}| + \|L\|)$.

The converse direction is not necessarily true. There does not seem to be a generic many-to-one reduction from the model checking problem for PDL[$\mathcal{L}$] to a single instance of the $\mathcal{L}$-reachability problem. However, a Turing reduction is possible. The following algorithm solves the model checking problem for PDL[$\mathcal{L}$]

given a procedure *Reach* which takes as arguments a graph $\mathcal{T}$, an $\mathcal{L}$-language, and a set U of target states and returns the set of all states which have an L-successor in U in this graph.

$$
\begin{array}{l}
\text{MC-PDL}(\mathcal{T},\, \varphi) = \\
\quad \texttt{let}\ (\mathcal{S}, \longrightarrow, \ell) = \mathcal{T}\ \texttt{in} \\
\quad \texttt{case}\ \varphi\ \text{of} \\
\qquad q \qquad\quad : \{s \in \mathcal{S} \mid q \in \ell(s)\} \\
\qquad \neg\psi \qquad : \mathcal{S} \setminus \text{MC-PDL}(\mathcal{T},\, \psi) \\
\qquad \psi_1 \vee \psi_2 : \text{MC-PDL}(\mathcal{T},\, \psi_1) \cup \text{MC-PDL}(\mathcal{T},\, \psi_2) \\
\qquad \langle L \rangle \psi \quad : \texttt{let}\ \ \longrightarrow' \,= \longrightarrow \cup\, \{(u, \psi?, u) \mid \psi \in \mathit{Tests}(L), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad u \in \text{MC-PDL}(\mathcal{T},\psi)\ \} \\
\qquad\qquad\quad \texttt{in}\ \mathit{Reach}((\mathcal{S}, \longrightarrow'),\, L,\, \text{MC-PDL}(\mathcal{T},\psi))
\end{array}
$$

Lemma 2. *Let $\mathcal{L}$ be a class of languages. The model checking problem for PDL[$\mathcal{L}$] Turing-reduces to the $\mathcal{L}$-reachability problem in quadratic time.*

Proof. It is not hard to see that algorithm MC-PDL can be made to run in time $\mathcal{O}(|\mathcal{T}| \cdot |\varphi|)$ when regarding *Reach* as an orcale. Using a dynamic programming approach one can restrict the numbers of recursive calls to one per subformula or test occurring in the input formula. Also, set operations and updates of the labeling function can be made to run in time $\mathcal{O}(|\mathcal{T}|)$. By assumption, $\mathit{Tests}(L)$ can be computed in time $\mathcal{O}(\|L\|)$ for every L occurring in φ.

Correctness of MC-PDL is straight-forward to show by induction on φ: for all states s of $\mathcal{T}$ we have: $s \in \text{MC-PDL}(\mathcal{T},\varphi)$ iff $\mathcal{T}, s \models \varphi$.

4.2 Forth and Back between Graphs and Formal Languages

Lemmas 1 and 2 provide a connection between $\mathcal{L}$-reachability and model checking PDL[$\mathcal{L}$]. This allows to transfer lower complexity bounds from the graph-theoretic side to the logical side, and upper complexity bounds vice-versa. We will provide a further link from the formal-language side, allowing to transfer complexity results from that side — which are usually easier to achieve than for the more specialised graph-theoretic or logical problems. This can also be viewed as the aim to find a sufficient and necessary condition on the class $\mathcal{L}$ of languages that guarantees the model checking problem for PDL[$\mathcal{L}$] to be decidable.

Lemma 3. *Let $\mathcal{L}$ be a class of languages. The REG-intersection problem for $\mathcal{L}$ reduces in linear time to the $\mathcal{L}$-reachability problem.*

Proof. Let $\mathcal{A}$ be an NFA $(Q, \Sigma, \delta, q_{\mathsf{init}}, F)$ and $L \in \mathcal{L}$. Define a graph $\mathcal{T}_\mathcal{A} := (Q, \rightarrow)$ with $s \xrightarrow{a} t$ iff $t \in \delta(s, a)$ for any $s, t \in Q$. Now note that $L \cap L(\mathcal{A}) \neq \emptyset$ iff there is a $f \in F$ s.t. $q_0 \xrightarrow{w} f$ for some $w \in L$ iff F is L-reachable from q_0 in $\mathcal{T}_\mathcal{A}$. Clearly, $|\mathcal{T}_\mathcal{A}| + \|L\| = \mathcal{O}(|\mathcal{A}| + \|L\|)$.

Lemma 4. *Let $\mathcal{L}$ be a class of languages. The $\mathcal{L}$-reachability problem reduces in linear time to the REG-intersection problem for $\mathcal{L}$.*

Proof. Let $\mathcal{T} = (\mathcal{S}, \to)$ be a graph, $s \in \mathcal{S}$, $T \subseteq \mathcal{S}$, and $L \in \mathcal{L}$. Define an NFA $\mathcal{A}_{\mathcal{T},s,T} := (\mathcal{S}, \Sigma, \delta, s, T)$ s.t. for all $t \in \mathcal{S}$ and all $a \in \Sigma$: $\delta(t,a) := \{u \mid t \xrightarrow{a} u\}$. Now there is a $w \in L$ and a $t \in T$ with $s \xrightarrow{w} t$ iff there is a path in $\mathcal{T}$ from s to some $t \in T$ s.t. the transition labels along that path form the word w. This is the case iff $w \in L(\mathcal{A}_{\mathcal{T},s,T}) \cap L$. Hence, there is such a w iff $L \cap L(\mathcal{A}_{\mathcal{T},s,T}) \neq \emptyset$. Clearly, $|\mathcal{A}_{\mathcal{T},s,T}| + \|L\| = \mathcal{O}(|\mathcal{T}| + \|L\|)$.

Theorem 1. *The model checking problem for PDL[$\mathcal{L}$] is equivalent under quadratic-time Turing reductions to the REG-intersection problem for $\mathcal{L}$.*

Proof. Immediately from Lemmas 1–4.

5 New Decidability and Complexity Results on Model Checking and Formal Language Constrained Reachability

Thm. 1 allows many known results from the theory of formal languages to be transfered to the model checking theory of PDL[$\mathcal{L}$]. For example, regular languages are closed under intersections and have a decidable non-emptiness problem. Hence, their REG-intersection problem is decidable, too. In fact, it is decidable in linear time which then yields quadratic-time decidability of the model checking problem for PDL[REG]. This has of course been known for a while [8].[3]

It is also known that CFL is closed under intersections with regular languages and has a non-emptiness problem that is decidable in polynomial time. Hence, Thm. 1 reproves that model checking for PDL[CFL] is also P-complete [18].

Note that a Turing reduction, i.e. an algorithm using an oracle an arbitrary number of times, is only needed in the embedding of the model checking problem into the REG-intersection problem. The other direction is realised as an ordinary reduction. Hence, complexity-theoretic hardness results can be transferred in this direction, too. For example, the class CSL of context-sensitive languages is closed under intersections with regular languages but its non-emptiness problem is undecidable. Hence, its REG-intersection problem is undecidable, too. The same holds for the classes ACFL of *alternating context-free languages* [21] and CL of *conjunctive languages* generated by *conjunctive grammars* [23,17]. Both extend the class of context-free languages by introducing conjunctions into context-free grammars. It then also holds for their extension *boolean grammars* which gives rise to *boolean languages* BL [24].

Corollary 1. *Let $\mathcal{L} \in \{\mathrm{CSL}, \mathrm{ACFL}, \mathrm{CL}, \mathrm{BL}\}$. Then the model checking problems for PDL[$\mathcal{L}$] as well as the $\mathcal{L}$-reachability problem are undecidable.*

[3] Note that PDL[REG] is often said to be model checkable in linear time. However, standard algorithms are only linear in the formula and in the structure but not in both.

Note that the non-emptiness problem for context-sensitive languages is r.e. because the word problem is decidable. However, since the reduction in Lemma 2 is only a Turing-reduction, recursive enumerability does not extend to the model checking problem. This would also contradict undecidability because model checking problems for logics like PDL are closed under complement. Thus, if it was r.e. it would also be co-r.e. and therefore decidable.

The limits of decidability lie somewhere between the context-free and the context-sensitive ones. One class of languages that is known to contain CFL and be contained in CSL itself is the class IL of *indexed languages* [1]. It is known that indexed languages are closed under intersections with regular languages (with polynomial blow-ups only) and that their non-emptiness problem is EXPTIME-complete [1,28]. Hence, so is their REG-intersection problem. Applying Thm. 1 again yields positive results for model checking and graph reachability.

Corollary 2. *The model checking problem for PDL[IL] and the IL-reachability problem are EXPTIME-complete.*

There are other classes which contain CFL, have a decidable non-emptiness problem and are closed under intersections with regular languages. For example, there are *mildly context-sensitive* formalisms like the class LIL of *linear indexed languages* [9,30]. Again, they are closed under intersections with regular languages and their non-emptiness problem is decidable — even in polynomial time. Since the blow-up in the construction of intersecting a linear-indexed grammar with a regular language is polynomial, their REG-intersection problem is PTIME-complete as well. Thm. 1 then transfers the upper bound to the corresponding model checking as well as graph reachability problem. A matching lower bound follows trivially from the model checking problem for PDL[REG] or PDL[CFL] for instance. In [30] it is shown that linear indexed grammars are equivalent under polynomial-time reductions to several other at first glance different formalisms, namely head grammars, combinatory categorical grammars and tree adjoining grammars. We denote by HL, CCL and TAL the language classes generated by those formalisms respectively.

Corollary 3. *Let* $\mathcal{L} \in \{\mathrm{LIL}, \mathrm{HL}, \mathrm{CCL}, \mathrm{TAL}\}$*. Then the model checking problem for PDL[*$\mathcal{L}$*] and the* $\mathcal{L}$*-reachability problem are PTIME-complete.*

Finally, another class of context-sensitive languages with nice algorithmic properties has recently been discovered: MVPL is the class of languages recognised by *multi-stack visibly pushdown languages* [29]. Since it is closed under intersections in general and subsumes REG it is closed under intersections with regular languages in particular. Furthermore, its emptiness problem is decidable in double exponential time. A lower bound is currently not known. Thm. 1 then transfers this upper bound to the model checking problem of PDL[MVPL].

Corollary 4. *The model checking problem for PDL[MVPL] and MVPL-reachability are decidable in 2EXPTIME.*

language class $\mathcal{L}$	REG-inters. for $\mathcal{L}$	$\mathcal{L}$-reachability	mod. check. PDL[$\mathcal{L}$]
ACFL, CL, BL, CSL	undec. [16]	undec. [5]	undec. (here)
MVPL	2EXPTIME [29]	2EXPTIME (here)	
IL	EXPTIME-c [1],[28]	EXPTIME-c (here)	
LIL, HL, CCL, TAL	PTIME-c [9],↓	PTIME-c (here)	
DCFL, CFL	PTIME-c [4],↓	PTIME-c [5],↓	PTIME-c [18],↓
SML, SSML, VPL	PTIME-c ↑,↓		
REG	NLOGSPACE-c [15]		PTIME-c [8], folkl.

Fig. 1. Decidability and complexity results for REG-intersection, reachability and model checking

6 Summary

Fig. 1 summarises the decidability and complexity results about the REG-intersection problem for class $\mathcal{L}$, the $\mathcal{L}$-reachability problem, and the model checking problem for PDL[$\mathcal{L}$] with regards to some of the most popular classes $\mathcal{L}$ between REG and CSL. For a complexity class $\mathcal{C}$ we write $\mathcal{C}$-c to denote completeness for this class under the usual reductions. The table contains references to the location where the results have been shown first. In case of completeness, if two references are given then the first one concerns the upper, the second one the lower bound. An arrow down states that the lower bound follows from the line below, an arrow up states that the upper bound follows from the line above. All the results of the two rightmost columns — apart from PTIME-hardness in the case of $\mathcal{L} = \text{REG}$ — can be derived from Thm. 1 and the REG-intersection column. The complexities in the last row of $\mathcal{L} = \text{REG}$ do not coincide as opposed to the other rows because NLOGSPACE is presumably not closed under quadratic-time reductions.

References

1. Aho, A.V.: Indexed grammars - an extension of context-free grammars. J. ACM 15(4), 647–671 (1968)
2. Alur, R., Madhusudan, P.: Visibly pushdown languages. In: Proc. 36th Ann. ACM Symp. on Theory of Computing (STOC 2004), pp. 202–211. ACM Press, New York (2004)
3. Baader, F., Lutz, C., Turhan, A.-Y.: Small is again beautiful in description logics. KI – Künstliche Intelligenz (2010) (to appear)
4. Bar-Hillel, Y., Perles, M., Shamir, E.: On formal properties of simple phrase structure grammars. Zeitschrift für Phonologie, Sprachwissenschaft und Kommunikationsforschung 14, 113–124 (1961)
5. Barrett, C., Jacob, R., Marathe, M.: Formal-language-constrained path problems. SIAM Journal on Computing 30(3), 809–837 (2000)

6. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
7. Fähndrich, M., Rehof, J.: Type-based flow analysis and context-free language reachability. Mathematical Structures in Computer Science 18(5), 823–894 (2008)
8. Fischer, M.J., Ladner, R.E.: Propositional dynamic logic of regular programs. Journal of Computer and System Sciences 18, 194–211 (1979)
9. Gazdar, G.: Applicability of indexed grammars to natural languages. In: Reyle, U., Rohrer, C. (eds.) Natural Language Parsing and Linguistic Theories, pp. 69–94. Reidel, Dordrecht (1988)
10. De Giacomo, G., Lenzerini, M.: Boosting the correspondence between description logics and propositional dynamic logics. In: Proc. of the 12th National Conference on Artificial Intelligence (AAAI 1994), pp. 205–212. AAAI-Press/The MIT-Press (1994)
11. Harel, D., Kaminsky, M.: Strengthened results on nonregular PDL. Technical Report MCS99-13, Weizmann Institute of Science, Faculty of Mathematics and Computer Science (1999)
12. Harel, D., Pnueli, A., Stavi, J.: Propositional dynamic logic of nonregular programs. Journal of Computer and System Sciences 26(2), 222–243 (1983)
13. Harel, D., Raz, D.: Deciding properties of nonregular programs. SIAM J. Comput. 22(4), 857–874 (1993)
14. Harel, D., Singerman, E.: More on nonregular PDL: Finite models and Fibonacci-like programs. Information and Computation 128(2), 109–118 (1996)
15. Hunt, H.B.: On the time and tape complexity of languages I. In: ACM (ed.) Conf. Rec. of 5th Annual ACM Symp. on Theory of Computing (STOC 1973), pp. 10–19. ACM Press, New York (1973)
16. Landweber, P.S.: Three theorems on phrase structure grammars of type 1. Inform. and Control 6, 131–136 (1963)
17. Lange, M.: Alternating context-free languages and linear time μ-calculus with sequential composition. In: Proc. 9th Workshop on Expressiveness in Concurrency (EXPRESS 2002). ENTCS, vol. 68.2, pp. 71–87. Elsevier, Amsterdam (2002)
18. Lange, M.: Model checking propositional dynamic logic with all extras. Journal of Applied Logic 4(1), 39–49 (2005)
19. Löding, C., Lutz, C., Serre, O.: Propositional dynamic logic with recursive programs. J. Log. Algebr. Program 73(1-2), 51–69 (2007)
20. Mehlhorn, K.: Pebbling mountain ranges and its application to DCFL-recognition. In: de Bakker, J.W., van Leeuwen, J. (eds.) ICALP 1980. LNCS, vol. 85, pp. 422–435. Springer, Heidelberg (1980)
21. Moriya, E.: A grammatical characterization of alternating pushdown automata. TCS 67(1), 75–85 (1989)
22. Nitta, N., Seki, H., Takata, Y.: Security verification of programs with stack inspection. In: SACMAT, pp. 31–40 (2001)
23. Okhotin, A.: Conjunctive grammars. Journal of Automata, Languages and Combinatorics 6(4), 519–535 (2001)
24. Okhotin, A.: Boolean grammars. Information and Computation 194(1), 19–48 (2004)
25. Prendinger, H., Schurz, G.: Reasoning about action and change. A dynamic logic approach. Journal of Logic, Language and Information 5(2), 209–245 (1996)
26. Reps, T.: Shape analysis as a generalized path problem. In: Proc. ACM SIGPLAN Symp. on Partial Evaluation and Semantics-Based Program Manipulation, pp. 1–11 (1995)

27. Reps, T.W.: Program analysis via graph reachability. Information & Software Technology 40(11-12), 701–726 (1998)
28. Tanaka, S., Kasai, T.: The emptiness problem for indexed language is exponential-time complete. Systems and Computers in Japan 17(9), 29–37 (2007)
29. La Torre, S., Madhusudan, P., Parlato, G.: A robust class of context-sensitive languages. In: Proc. 22nd Conf. on Logic in Computer Science (LICS 2007), pp. 161–170. IEEE, Los Alamitos (2007)
30. Vijay-Shanker, K., Weir, D.J.: The equivalence of four extensions of context-free grammars. Mathematical Systems Theory 27, 27–511 (1994)