Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling

Automatic Reductions from PH into STRIPS or How to Generate Short Problems with Very Long Solutions

Aldo Porco
Departamento de Computación
Universidad Simón Bolívar
Caracas, Venezuela
aldo@gia.usb.ve

Alejandro Machado
Departamento de Computación
Universidad Simón Bolívar
Caracas, Venezuela
alejandro@gia.usb.ve

Blai Bonet
Departamento de Computación
Universidad Simón Bolívar
Caracas, Venezuela
bonet@ldc.usb.ve

Abstract

Recently, it has been shown how to automatically translate any problem in NP, expressed in the language of second-order logic, into a STRIPS planning problem. In this work, we extend this translation by considering decision problems in the polynomial-time hierarchy (PH) and not just NP. Since decision problems in PH require in general exponentially-long "certificates", the plans (if any) for the resulting STRIPS problems may have exponential length. Besides explaining the novel translations, we present experimental results and discuss the challenges that such problems pose.

Introduction

In a recent contribution, Porco, Machado, and Bonet (2011) show how to automatically reduce instances of decision problems in NP into plan-existence decision problems for STRIPS. In this approach, a decision problem Π in NP is encoded as a second-order existential (SO$\exists$) formula Φ, while problem instances are encoded as first-order structures $\mathcal{A}$ such that $\mathcal{A} \models \Phi$ iff the instance encoded by $\mathcal{A}$ belongs to Π. This is a general and feasible method as it is known that SO$\exists$ captures the class NP (Immerman 1998). The pair $\langle \Phi, \mathcal{A} \rangle$ is then fed into a software tool that outputs a STRIPS problem P that has a solution (plan) iff $\mathcal{A} \models \Phi$, thus allowing the utilization of the current planning technology to automatically solve the problems in NP that are expressed as second-order formulas. Moreover, the problem P is not an arbitrary STRIPS problem but one that can be solved in polytime by a non-deterministic Turing machine, thus guaranteeing that the whole approach is not an overkill, from the standpoint of complexity theory. Furthermore, the function $\langle \Phi, \cdot \rangle$: STRUC $\mapsto$ STRIPS that is implemented by this tool for fixed Φ, and that maps first-order structures $\mathcal{A}$ into STRIPS problems P, is computable in polynomial time and thus corresponds to a genuine *polytime many-one reduction*.

Since it is known that other (bigger) complexity classes are also captured in second-order logic, the development of a tool capable of targeting these classes was left as an open issue. In this paper, we bridge this gap and show in a concise and crisp manner how the tool can be extended to work with decision problems in the Polynomial-time Hierarchy (PH) (Sipser 2005). However, although the resulting reductions still run in polynomial time, the resulting STRIPS problems are not solvable in non-deterministic polynomial time. This is so because these problems only admit plans of *exponential length*, in the worst case, since PH contains, besides NP, the classes coNP and Σ^p_k for any $k \geq 1$. The resulting planning problems are thus short, meaning that they can be encoded by short bit sequences, but have long solutions.

The consequences of having small problems with long solutions are several. Among others, such problems pose a challenge on the current state-of-the-art planners, which are based on heuristic search. In particular, heuristics functions that are based on the delete relaxation or that do not take into account that the same action may be applied a large (i.e., exponential) number of times will probably fail at being effective for guiding the search. It is not surprising though that the current best SAT-based planners, the planners M and Mp, are able to solve some of these challenging STRIPS problems for which forward-search planners like LAMA'11 are totally lost. Yet, surprisingly, LAMA'11 is quite superior to M in one of the tested domains that requires incredible long plans for some instances.

Being this a short paper, we don't have the space to present a detailed account of the approach by Porco, Machado, and Bonet (2011). Instead, we revise some known results from Descriptive Complexity, sketch how problems in PH can be automatically translated into STRIPS, and present experimental results.

Capturing PH in Second-Order Logic

NP is captured by SO$\exists$ (Fagin 1974); i.e., decision problems Π in NP are characterized by second-order formulas Φ like

$$\Phi = (\exists R_1^{a_1}) \cdots (\exists R_n^{a_n}) \psi \,, \tag{1}$$

where each symbol $R_i^{a_i}$ is a existentially-quantified relation of arity a_i, and ψ is an arbitrary first-order sentence over a vocabulary containing $\{R_i^{a_i}\}_{i=1}^n$. In this view, Π is the set of instances (words in a language) that when encoded as first-order structures $\mathcal{A}$ satisfy the formula Φ. Likewise, coNP is captured by SO$\forall$ that contains formulas like Eq. (1) but with the second-order existential quantifiers replaced by second-order universal quantifiers.

In general, Σ^p_k is captured by formulas whose second-order quantifiers show a structure of k alternating blocks

Copyright © 2013, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

of quantifiers, beginning with existentials. For example, $\Sigma^p_2 = \text{SO}\exists\forall$ corresponds to formulas of the form:

$$\Phi = (\exists R_1^{a_1}) \cdots (\exists R_n^{a_n})(\forall S_1^{a'_1}) \cdots (\forall S_m^{a'_m})\psi\,. \qquad (2)$$

Since PH equals $\bigcup_{k\geq 1} \Sigma^p_k$ and is fully characterized by arbitrary second-order formulas (Immerman 1998), the new tool must target such formulas instead of the more restricted SO$\exists$ formulas. On the other hand, PH $\subseteq$ PSPACE, the decision problem for STRIPS is PSPACE-complete and there are no known syntactic restrictions on STRIPS that place its plan-existence decision problem in PH. Thus, the new tool must generate unrestricted STRIPS problems.

Consider now the formula $\Phi = (\forall R^1)\psi$ and structure $\mathcal{A}$ with universe $|\mathcal{A}|$ (i.e., $|\mathcal{A}|$ is a symbol that denotes the set of objects in $\mathcal{A}$). This formula is valid in $\mathcal{A}$ iff for *every interpretation* $R^{\mathcal{A}}$ of R, where $R^{\mathcal{A}} \subseteq |\mathcal{A}|$, we have $\langle \mathcal{A}, R^{\mathcal{A}}\rangle \vDash \psi$. Thus, since there are $2^{\|\mathcal{A}\|}$ different interpretations for R, a proof for $\mathcal{A} \vDash \Phi$ may be of *exponential length*. For example, if Φ_{UNSAT} denotes UNSAT, then a proof that a structure $\mathcal{A}$ (encoding a set of clauses) is unsatisfiable is typically of exponential length as it must consider all the truth valuations for the propositional variables in $\mathcal{A}$. Indeed, the solutions for the planning problem P generated by the pair $\langle \Phi_{\text{UNSAT}}, \mathcal{A}\rangle$ encode such proofs and are of exponential length.

Automatic Translations into STRIPS

Before extending our previous work over PH, we found it convenient to add a simple *type system*. We refer to a type system τ^* constructed from a finite set $\tau = \{\mathbf{t}_0, \mathbf{t}_1, \ldots, \mathbf{t}_N\}$ of *atomic types*, and where τ^* refers to the smallest set that contains τ and all types of the form $t = t' \times t''$ for $t' \in \tau^*$ and $t'' \in \tau$. The idea is that each object in a first-order structure is assigned an atomic type, with $\mathbf{t}_0$ standing for the type containing all objects. The types are then used to qualify the first-order and second-order quantifications in formulas. For example, $(\forall x \in t)$ refers to a first-order quantification over all objects of type $t \in \tau$, while $(\forall R^t)$ refers to a second-order universally quantified predicate R with type $t \in \tau^*$. The only restriction that we pose is that the complex types in $\tau^* \setminus \tau$ should only appear in second-order quantifications.

Let us consider a general SO formula of the form

$$\Phi = (\mathcal{Q}_1 R_1^{t_1})(\mathcal{Q}_2 R_2^{t_2}) \cdots (\mathcal{Q}_n R_n^{t_n})\psi \qquad (3)$$

where each $\mathcal{Q}_i \in \{\exists, \forall\}$ is a second-order quantifier, R_i is a relational symbol of type $t_i \in \tau^*$, and ψ is a first-order sentence. Further, let $\mathcal{A}$ be a first-order structure over the vocabulary of ψ. We now show how to generate a planning problem P that has solution iff $\mathcal{A} \vDash \Phi$.

Following the previous work, the problem P has atom schemas Holds-$\mathfrak{F}[\theta]$ for every subformula θ of ψ, with parameters that match the free variables in θ. For example, for the formula Φ_{SAT} encoding SAT, given by

$$\begin{aligned}\Phi_{\text{SAT}} &= (\exists T^{\text{Var}})\psi_{\text{SAT}}\,,\\ \psi_{\text{SAT}} &= (\forall y \in \text{Cls})(\exists x \in \text{Var})\\ &\qquad [(P(x,y) \wedge T(x)) \vee (N(x,y) \vee \neg T(x))]\end{aligned}$$

where 'Var' and 'Cls' are the types for variables and clauses, there is an atom Holds-$\mathfrak{F}[\theta]$ with parameters $\langle x, y\rangle$ for the subformula $\theta(x,y) = N(x,y) \vee \neg T(x)$. In ψ_{SAT}, the relation $P(x,y)$ (resp. $N(x,y)$) denotes that the variable x appears positively (resp. negatively) in the clause y; these relations are fixed for a given SAT instance and their interpretation given in a structure $\mathcal{A}$ encoding the instance. On the other hand, the existentially-quantified relation T encodes the model: $T(x)$ is true iff the variable x is assigned to true.

The problem P has actions for choosing the valuation T, that is represented by fluents of the form '$\texttt{T}(?x)$' and '$\texttt{not\_T}(?x)$', and actions for building a proof for ψ_{SAT}, that are designed to work by following the recursive structure of ψ_{SAT}. The goal of P is the single fluent Holds-$\mathfrak{F}[\psi_{\text{SAT}}]$ that has no parameters since ψ_{SAT} is a sentence.

In our approach, we make use of the same fluents that denote the validity of subformulas and the quantified relations. The difference, though, is in how the quantified relations are built and *interleaved* with the proofs of the subformulas. The interleaving issue does not arise for SO$\exists$ as there is just one interpretation to construct for each existentially-quantified relation, but for general formulas, one needs to construct and test many different interpretations. To make this point clear, let us consider the formula for UNSAT that involves a universally-quantified unary relation T:

$$\Phi_{\text{UNSAT}} = (\forall T^{\text{Var}})\psi_{\text{UNSAT}} \quad \text{with} \quad \psi_{\text{UNSAT}} \equiv \neg\psi_{\text{SAT}}\,.$$

This formula says that for every relation T over variables (that encodes a model), there is a clause y such that for every variable x, if x appears positive in y, then x is false, and if x appears negative in y, x is true. UNSAT can thus be automatically translated into STRIPS by considering actions that "iterate" over all possible relations T, and actions for obtaining the fluent Holds-$\mathfrak{F}[\psi_{\text{UNSAT}}]$ for each such T.

For unary T, there are 2^n different relations on n variables. Each relation can be encoded with a binary string of length n (one bit per variable) so that the ith bit is 1 iff the ith variable x_i is true. Thus, to iterate over all relations is equivalent to iterate over all such strings. This can be done by starting with the empty relation, corresponding to '0…00', and successively "adding 1" until reaching '1…11'. Figure 1 shows 5 actions that do the iteration; these actions make use of the extra fluents Need- and Holds-$\mathfrak{F}[\Phi_{\text{UNSAT}}]$, for the second-order formula Φ_{UNSAT}, and the fluents 'Marker(x)', 'VarFirst(x)', 'VarSucc(x,y)' and 'VarLast(x)'. The last three type of fluents are *static* fluents that are set in the initial situation and implement an static order of the objects that refer to propositional variables.

Let us briefly explain how the iteration works. The initial situation contains the static fluents defining the Var and Cls types, and the binary relations $N(x,y)$ and $P(x,y)$, plus the fluent Need-$\mathfrak{F}[\Phi_{\text{UNSAT}}]$, while the goal contains only Holds-$\mathfrak{F}[\Phi_{\text{UNSAT}}]$. Initially, the only applicable action is A1, which sets T as the empty relation and adds Need-$\mathfrak{F}[\psi_{\text{UNSAT}}]$. Although not shown, this fluent triggers actions for achieving a proof of ψ_{UNSAT} for the current T, which as a side effect add Holds-$\mathfrak{F}[\psi_{\text{UNSAT}}]$ once ψ_{UNSAT} is proved. If ψ_{UNSAT} is proved, A2 becomes the only applicable operator and it removes Holds-$\mathfrak{F}[\psi_{\text{UNSAT}}]$ and adds Marker(x) for the first variable

[A1] StartProof:
Pre: Need-$\mathfrak{F}[\Phi_{\text{UNSAT}}]$
Eff: for each $x \in$ Var: not-T(x) ,
$\neg$Need-$\mathfrak{F}[\Phi_{\text{UNSAT}}]$, Need-$\mathfrak{F}[\psi_{\text{UNSAT}}]$

[A2] NextIterate(x):
Pre: Holds-$\mathfrak{F}[\psi_{\text{UNSAT}}]$, VarFirst(x)
Eff: $\neg$Holds-$\mathfrak{F}[\psi_{\text{UNSAT}}]$, Marker(x)

[A3] NextOverExcludedObject(x):
Pre: Marker(x) , not-T(x)
Eff: $\neg$not-T(x) , T(x) , $\neg$Marker(x) , Need-$\mathfrak{F}[\psi_{\text{UNSAT}}]$

[A4] NextOverIncludedObject(x, y):
Pre: Marker(x) , T(x) , VarSucc(x, y)
Eff: $\neg$T(x) , not-T(x) , $\neg$Marker(x) , Marker(y)

[A5] FinishProof(x):
Pre: Marker(x) , T(x) , VarLast(x)
Eff: $\neg$T(x) , not-T(x) , $\neg$Marker(x) , Holds-$\mathfrak{F}[\Phi_{\text{UNSAT}}]$

Figure 1: Actions for iterating over the 2^n unary relations T that encode the truth-assignments for n propositions in UNSAT.

x in the static order. After A2, A3 must be applied and it changes T from the model corresponding to $0\ldots00$ to the model corresponding to $0\ldots01$, and adds Need-$\mathfrak{F}[\psi_{\text{UNSAT}}]$ that asks for a proof of ψ_{UNSAT} for the new T. This iteration proceeds in a similar manner until proving ψ_{UNSAT} for the last model corresponding to $1\ldots11$, a time at which A5 becomes the only applicable action and adds Holds-$\mathfrak{F}[\Phi_{\text{UNSAT}}]$.[1]

This iteration also works for relations of arity $k > 1$. The only requirement is to replace x and y with k-tuples, and have fluents that implement the static order over k-tuples.

In the case of SAT, the new translation differs from the old by having fluents for the SO formula Φ_{SAT} and by "passing control" to the actions that prove Holds-$\mathfrak{F}[\psi_{\text{SAT}}]$ in a way similar to UNSAT; Fig. 2 shows the operators for SAT.

The translations for SAT and UNSAT share a common "protocol" that controls which operators become active or inactive depending on which part of the formula is being proved. This protocol is designed to allow the composition of the two types of SO quantifiers, and thus to make a translation for PH. For example, a formula in Σ_2^p of the form $\Phi = (\exists T^{t_1})(\forall R^{t_2})\psi$ can be decomposed as $\Phi = (\exists T^{t_1})\Psi$ with $\Psi = (\forall R^{t_2})\psi$, so that operators like the ones for SAT, with fluents Need- and Holds-$\mathfrak{F}[\Psi]$, may be combined with operators like the ones for UNSAT with fluents Need- and Holds-$\mathfrak{F}[\psi]$. In such combination, the E4 operator for the inner SO existential must delete the fluents for the subformulas in order to have a "fresh state" for the next proof.

Horizon Windows for SAT-based Planners

Given an SO formula Φ describing a property Π and a structure $\mathcal{A}$ encoding an instance of the problem, one wants to decide whether or not the instance satisfies the property. The translation generates in polytime (in the size of $\mathcal{A}$) a STRIPS problem P that has solution iff $\mathcal{A} \models \Phi$. The problem P thus may be solved with any complete planner to answer the original question. However, SAT-based planners are inherently incomplete when there is no solution, as they continue the search forever unless an upper bound on the length of the plan is given in advance.[2]

[1]The fluents Marker, etc. are for the SO quantifier $\forall T$. When composing translations, each SO quantifier has its own fluents.

[E1] StartProof:
Pre: Need-$\mathfrak{F}[\Phi_{\text{SAT}}]$
Eff: for each $x \in$ Var: not-T(x), $\neg$T(x) , $\neg$Need-$\mathfrak{F}[\Phi_{\text{SAT}}]$, Guess

[E2] SetTrue(x):
Pre: Guess ; Eff: T(x) , $\neg$not-T(x)

[E3] ProofSubformula:
Pre: Guess ; Eff: $\neg$Guess , Need-$\mathfrak{F}[\psi_{\text{SAT}}]$

[E4] FinishProof:
Pre: Holds-$\mathfrak{F}[\psi_{\text{SAT}}]$; Eff: $\neg$Holds-$\mathfrak{F}[\psi_{\text{SAT}}]$, Holds-$\mathfrak{F}[\Phi_{\text{SAT}}]$

Figure 2: Actions that implement the second-order existential quantifier in SAT.

Porco, Machado, and Bonet (2011) show how to calculate *tight* lower and upper bounds $l(\psi)$ and $u(\psi)$ on the number of (parallel) actions needed to prove a *first-order* sentence ψ once the interpretations of the relational symbols occurring in ψ are fixed. We use such estimates to provide tight lower and upper bounds $l^*(\Phi)$ and $u^*(\Phi)$ on the length of *parallel plans* for SO formulas Φ. We proceed inductively:

1. For first-order ψ, $l^*(\psi) = l(\psi)$ and $u^*(\psi) = u(\psi)$.
2. If $\Phi = (\exists R^t)\Psi$, then $l^*(\Phi) = 3 + l^*(\Psi)$ and $u^*(\Phi) = 4 + u^*(\Psi)$ since operators E1, E3 and E4 always need to be applied. If E2 is required, all applications can be performed in one parallel step. Thus, if one dummy NoOp action E5 is added, the lower bound $l^*(\Phi)$ can be pushed by 1 unit to match the upper bound $u^*(\Phi)$.
3. If $\Phi = (\forall R^t)\Psi$, then
$$l^*(\Phi) = 2^k l^*(\Psi) + 2^k + 2^{k+1} = 2^k(3 + l^*(\Psi))\,,$$
and also for $u^*(\Phi)$, where k is the number of objects of type t. This is so because A1 and A5 are executed, 2^k proofs of Ψ are required, and after each such proof A2 is executed, and the number of bits flipped when incrementing a k-bit counter from zero to its maximum is $2^{k+1} - 2$ (Cormen, Leiserson, and Rivest 1990).

Experiments

We performed experiments on domains encoding different propositional QBFs in CNF format, and on domains encoding non-3-colorability problems on graphs (denoted by $\overline{\text{3Col}}$). The QBFs span initial levels of PH, while $\overline{\text{3Col}}$ is coNP-complete. For example, we considered $\forall\exists$-QBFs encoded with the SO formula

$$(\forall T_1^{\text{V}_1})(\exists T_2^{\text{V}_2})(\forall y \in \text{Cls})(\exists x \in \text{Var})$$
$$\big[P(x,y) \wedge [(\text{V}_1(x) \wedge T_1(x)) \vee (\text{V}_2(x) \wedge T_2(x))]\big] \vee$$
$$\big[N(x,y) \wedge [(\text{V}_1(x) \wedge \neg T_1(x)) \vee (\text{V}_2(x) \wedge \neg T_2(x))]\big]$$

[2]If P has n atoms, 2^n is a trivial and non-informative bound.

∃∀	#∃	#∀		n	+	−	time	len	PDDL
	60	1		5	—	5	184.3	—	18.4
		2		5	—	1	4,382.5	—	18.5
	100	1		5	4	1	176.6	316	20.4
		2		5	3	2	3,471.9	628	20.5
∃∀∃	#∃	#∀	#∃	n	+	−	time	len	PDDL
	10	2	30	5	—	5	4,199.2	—	17.5
		2	50	5	—	5	2,313.9	—	18.4
	30	2	30	5	—	5	3,210.7	—	18.5
		2	50	5	—	5	3,166.3	—	19.4
	50	2	30	5	—	1	3,313.4	—	19.4
		2	50	5	3	2	3,450.9	640	20.4
∀∃	#∀	#∃		n	+	−	time	len	PDDL
	1	100		5	5	—	147.9	319	20.2
	2	30		5	—	5	2,060.4	—	16.9
		60		5	4	1	3,100.7	637	18.4
		80		5	5	—	2,893.2	637	19.3
		100		5	5	—	2,092.8	637	20.2
	3	15		5	—	—	—	—	—
∀∃∀	#∀	#∃	#∀	n	+	−	time	len	PDDL
	1	60	1	5	1	—	404.7	635	21.1
		60	2	5	—	—	—	—	21.2
		80	1	5	5	—	403.2	635	22.0
		100	1	5	5	—	390.8	635	23.1
	2	60	1	5	2	—	456.7	1,269	21.2
		60	2	5	—	—	—	—	21.3
		80	1	5	5	—	499.5	1,269	22.1
		100	1	5	5	—	454.9	1,269	23.2

Table 1: Random QBF problems with 150 clauses each and 4 types of QBFs: ∃∀, ∃∀∃, ∀∃ and ∀∃∀. Columns for number of variables of each type, number of instances (n), number of positive ($+$) and negative ($-$) instances, and averages for time (in seconds), *parallel* plan length (for solved instances), and PDDL size (in KB).

where V_1 and V_2 are types that partition the variables into universal and existential. Due to lack of space, we only present results in Table 1 that correspond to random CNFs of 150 clauses,[3] 3-CNF for ∃∀, ∀∃ and ∃∀∃ and 4-CNF for ∀∃∀ instances. All instances were solved with M (Rintanen 2010), and verified with a QBF solver, on a 2.33 GHz 5140 Xeon CPU with 2GB of RAM and 1.5 hours cutoff time. The benchmark contains positive and negative instances.

SAT-based planners do not perform well in $\overline{\text{3Col}}$. Thus, we tried the state-of-the-art search-based planner LAMA'11 (Richter and Westphal 2010) for random $G(n, p)$ graphs (Bollobás 2001), for $n \in \{4, 5, \ldots, 9\}$ and $p \in \{\frac{1}{5}, \frac{2}{5}, \ldots, 1\}$, on the same machine and with the same constraints on time and space. The results and statistics for the average plan length and encoding size appear in Table 2. Surprisingly, LAMA'11 is able to find very long plans in short time; e.g., for graphs with 7 and 8 vertices, LAMA'11 found plans with more than 100,000 and 400,000 actions respectively, in a few minutes. We believe that the success of LAMA'11 for these instances is mainly due to the *implicit serialization* of subgoals that results from the utilization of multiple (open) queues with different heuristics (Richter and Westphal 2010; Lipovetzky and Geffner 2012). Certainly, these problems are not trivial for search-based planners as shown in Table 3, where A* with a zero heuristic was used.

[3]Generated with BLOCKSQBF (http://fmv.jku.at/blocksqbf) that is based on the random model of Chen and Interian (2005).

V	n	+	−	time / +	time / −	plan len	PDDL
4	5	1	4	0.1	0.8	1,731	0.4
5	5	2	3	0.6	67.9	6,695	0.6
6	5	2	3	3.4	464.9	26,163	0.7
7	5	2	2	74.8	1.6	102,935	0.8
8	5	1	2	624.0	5.9	406,851	1.0
9	5	—	1	—	0.3	—	1.1

Table 2: LAMA'11 on random $\overline{\text{3Col}}$ instances. Columns for number of vertices ($|V|$), number of instances (n) and solved instances ($+$ and $-$), and averages for time (in seconds, for $+$ and $-$ instances), plan length (for $+$ instances), and PDDL size (in KB).

V	n	+	−	time / +	time / −	plan len	PDDL
4	5	1	4	1,850.1	0.1	1,731	0.4
5	5	—	3	—	11.7	—	0.6
6	5	—	3	—	81.9	—	0.7
7	5	—	2	—	0.2	—	0.8
8	5	—	2	—	1.0	—	1.0
9	5	—	1	—	0.0	—	1.1

Table 3: Blind search on random $\overline{\text{3Col}}$ instances. Columns for number of vertices ($|V|$), number of instances (n) and solved instances ($+$ and $-$), and averages for time (in seconds, for $+$ and $-$ instances), plan length (for $+$ instances), and PDDL size (in KB).

Discussion

We have extended the tool by Porco, Machado, and Bonet (2011) with a type system and over arbitrary SO formulas which capture the class PH: given such a formula Φ over signature σ and a first-order structure $\mathcal{A}$ for σ, the tool generates a planning problem P that has solution iff $\mathcal{A} \models \Phi$. Since problems in PH don't have short certificates in general, the solutions for STRIPS problems generally have exponential length. Thus, we do not expect that planners based on forward search in state space will be able to succeed unless heuristics that account for the multiple application of actions are used. However, the results for LAMA'11 on $\overline{\text{3Col}}$ contradict this expectation and thus deserve an explanation. We conjecture that the reason for this success is the implicit serialization of subgoals, but further investigation is needed.

The extension over formulas that capture PSPACE is now foreseeable, as PSPACE equals SO plus transitive closure (TC) (Immerman 1998), and TC is the computation of a connectivity relation in an implicit graph which is a standard practice in planning and STRIPS.

Acknowledgements

Experiments done on the Orion Cluster at the Departamento de Tecnología, Universitat Pompeu Fabra, Barcelona, Spain.

References

Bollobás, B. 2001. *Random Graphs*. Cambridge University Press, second edition.

Chen, H., and Interian, Y. 2005. A model for generating random quantified boolean formulas. In *Proc. 19th Int. Joint Conf. on Artificial Intelligence*, 66–71.

Cormen, T.; Leiserson, C.; and Rivest, R. 1990. *Introduction to Algorithms*. MIT Press.

Fagin, R. 1974. Generalized first-order spectra and polynomial-time recognizable sets. *American Mathematical Society* 7:27–41.

Immerman, N. 1998. *Descriptive Complexity*. Springer.

Lipovetzky, N., and Geffner, H. 2012. Width and serialization of classical planning problems. In *Proc. 20th European Conf. on Artificial Intelligence*, 540–545.

Porco, A.; Machado, A.; and Bonet, B. 2011. Automatic polytime reductions of NP problems into a fragment of STRIPS. In *Proc. 21st Int. Conf. on Automated Planning and Scheduling*, 178–185.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39(1):127–177.

Rintanen, J. 2010. Heuristics for planning with SAT. In Cohen, D., ed., *Proc. 16th Int. Conf. on Principles and Practice of Constraint Programming*, 414–428. St. Andrews, Scotland: Springer: LNCS 6308.

Sipser, M. 2005. *Introduction to Theory of Computation, 2nd Edition*. Boston, MA: Thomson Course Technology.