A Computational Perspective on Fragments of the Dynamic Logic of Propositional Assignments

Abdallah Saffidine
UNSW Sydney, Australia
(joint work with Andreas Herzig)

DIMAP Seminar
June 21, 2021

A Bird's eye view

Introducing DLPA

- Warm-up with Propositional logic and QBF
- DLPA Syntax and Semantics
- Modeling problems in DLPA

Comparing Logics: Expressivity, Complexity, Succinctness

- Propositional logic warm-up.
- Known results relating DLPA and QBF.
- DLPA is no more succinct than QBF (new).

Comput. Complexity: Satisfiability, Validity, Model Checking

- Propositional logic warm-up.
- QBF and the polynomial hierarchy.
- Connecting DLPA fragments to the poly. hierarchy (new).

Outline

Interpretation and Models

Propositional Formulas (PROP)

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \rightarrow \varphi) \qquad p \in \mathbb{P}$$

Definition

- A *valuation* or *model* is a subset of $\mathbb{P}$: those p assigned true.
- A model $V \subseteq \mathbb{P}$ satisfies φ if "it makes φ true". We write $V \models \varphi$.
- The *interpretation* of a formula φ is
  the set of models $||\varphi|| \subseteq 2^{\mathbb{P}}$ that satisfy φ.

Example: $\varphi = a \vee (\neg b \wedge c)$

Does the model ... satisfy φ?

$\{a\}$	Yes
$\{b\}$	No
$\{c\}$	Yes
$\{a, b, c\}$	Yes

Quantified Boolean Formulas

Grammar / Syntax

$$\mathcal{L}_{\text{QBF}} : \varphi \quad ::= \quad p \mid \top \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists P \varphi$$

where p ranges over $\mathbb{P}$ and P over the set of finite subsets of $\mathbb{P}$.

Interpretation / Semantics

$$\|\top\| = 2^{\mathbb{P}}; \qquad \|p\| = \{v : p \in v\},$$

$$\|\varphi \vee \psi\| = \|\varphi\| \cup \|\psi\|; \qquad \|\neg\varphi\| = 2^{\mathbb{P}} \setminus \|\varphi\|;$$

$$\|\exists P \varphi\| = \{v : \text{a partition } Q^+ \dot\cup Q^- = P \text{ s.t. } v \cup Q^+ \setminus Q^- \in \|\varphi\|\}$$

Example: $\varphi = \exists\{a\}\forall\{b\}(b \leftrightarrow c) \vee ((a \leftrightarrow d) \wedge (b \leftrightarrow d))$

Does the model	$\{c\}$	satisfy φ?	Yes
	$\{c, d\}$		No
	$\{a, b, c\}$		Yes
	$\{a, d\}$		Yes

Language of DL-PA: Syntax

Grammar

Complex formulas and complex programs:

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \vee \varphi) \mid \langle\pi\rangle\varphi$$

$$\pi ::= (p{\leftarrow}\varphi) \mid \varphi? \mid (\pi;\pi) \mid (\pi \cup \pi) \mid \pi^*$$

for $p \in \mathbb{P}$ (set of propositional variables)

Intuition

$\langle\pi\rangle\varphi$ and $[\pi]\varphi \stackrel{\text{def}}{=} \neg\langle\pi\rangle\neg\varphi$ are existential/universal modal operators.

$\langle\pi\rangle\varphi$	=	"φ is true after *some* execution of π"
$[\pi]\varphi$	=	"φ is true after *every* execution of π"
$p{\leftarrow}\varphi$	=	update p to $\top$ or $\bot$ depending on whether φ holds
$\varphi?$	=	fail if φ doesn't hold, otherwise proceed
$\pi_1;\pi_2$	=	execute π_1 then execute π_2
$\pi_1 \cup \pi_2$	=	execute π_1 or execute π_2
π^*	=	execute π some number of times

Programs as relations on valuations

- assignment:

$$v \stackrel{p \leftarrow \varphi}{\longrightarrow} v \cup \{p\} \quad \text{if } v \models \varphi$$
$$v \stackrel{p \leftarrow \varphi}{\longrightarrow} v \setminus \{p\} \quad \text{if } v \not\models \varphi$$

- test:

$$v \stackrel{\varphi?}{\longrightarrow} v' \ \text{ iff } \ v = v' \text{ and } v \models \varphi$$

- sequential composition:

$$v_1 \stackrel{\pi_1;\pi_2}{\longrightarrow} v_3 \ \text{ iff } \ \text{there is } v_2 \text{ such that } v_1 \stackrel{\pi_1}{\longrightarrow} v_2 \stackrel{\pi_2}{\longrightarrow} v_3$$

- nondeterministic composition:

$$v \stackrel{\pi_1 \cup \pi_2}{\longrightarrow} v' \ \text{ iff } \ v \stackrel{\pi_1}{\longrightarrow} v' \text{ or } v \stackrel{\pi_2}{\longrightarrow} v'$$

- finite iteration ('Kleene star'):

$$v \stackrel{\pi^*}{\longrightarrow} v' \ \text{ iff } \ \text{there is } n \text{ such that } v \Big(\stackrel{\pi}{\longrightarrow} \Big)^n v'$$

- write $(v, v') \in ||\pi||$ instead of $v \stackrel{\pi}{\longrightarrow} v'$

Programs as relations on valuations

- assignment:

$$v \xrightarrow{p \leftarrow \varphi} v \cup \{p\} \quad \text{if } v \models \varphi$$
$$v \xrightarrow{p \leftarrow \varphi} v \setminus \{p\} \quad \text{if } v \not\models \varphi$$

- test:

$$v \xrightarrow{\varphi?} v' \ \text{ iff } \ v = v' \text{ and } v \models \varphi$$

- sequential composition:

$$v_1 \xrightarrow{\pi_1;\pi_2} v_3 \ \text{ iff } \ \text{there is } v_2 \text{ such that } v_1 \xrightarrow{\pi_1} v_2 \xrightarrow{\pi_2} v_3$$

- nondeterministic composition:

$$v \xrightarrow{\pi_1 \cup \pi_2} v' \ \text{ iff } \ v \xrightarrow{\pi_1} v' \text{ or } v \xrightarrow{\pi_2} v'$$

- finite iteration (‘Kleene star’):

$$v \xrightarrow{\pi^*} v' \ \text{ iff } \ \text{there is } n \text{ such that } v\Big(\xrightarrow{\pi}\Big)^n v'$$

- write $(v, v') \in ||\pi||$ instead of $v \xrightarrow{\pi} v'$

Semantics of DL-PA: Interpretation

$\| \cdot \|$ for formulas: set of valuations $\subseteq 2^{\mathbb{P}}$

$$\|p\| = \{v \ : \ p \in v\}$$

$$\|\neg\varphi\| = 2^{\mathbb{P}} \setminus \|\varphi\|$$

$$\|\varphi \vee \psi\| = \|\varphi\| \cup \|\psi\|$$

$$\|\langle\pi\rangle\varphi\| = \Big\{v \ : \ \text{there is } v' \text{ such that } (v, v') \in \|\pi\| \ \ \& \ \ v' \in \|\varphi\|\Big\}$$

$\| \cdot \|$ for programs: binary relation on the set of valuations $2^{\mathbb{P}}$

$$\|p{\leftarrow}\varphi\| = \Big\{(v, v \cup \{p\}) \ : \ v \in \|\varphi\|\Big\} \cup \Big\{(v, v \setminus \{p\}) \ : \ v \notin \|\varphi\|\Big\}$$

$$\|\varphi?\| = \{(v, v) \ : \ v \in \|\varphi\|\}$$

$$\|\pi; \pi'\| = \|\pi\| \circ \|\pi'\|$$

$$\|\pi \cup \pi'\| = \|\pi\| \cup \|\pi'\|$$

$$\|\pi^*\| = (\|\pi\|)^* = \bigcup_{k \in \mathbb{N}_0} (\|\pi\|)^k$$

Reminder

$\langle\pi\rangle\varphi$	=	"φ is true after *some* execution of π"
$p \leftarrow \varphi$	=	update p to $\top$ or $\bot$ depending on whether φ holds
$\varphi?$	=	fail if φ doesn't hold, otherwise proceed
$\pi_1 ; \pi_2$	=	execute π_1 then execute π_2
$\pi_1 \cup \pi_2$	=	execute π_1 or execute π_2
π^*	=	execute π some number of times

Example: $\varphi = \exists\{a\}\forall\{b\}(b \leftrightarrow c) \vee ((a \leftrightarrow d) \wedge (b \leftrightarrow d))$

	Does the model			
Satisfy the formula	$\{\}$	$\{a\}$	$\{b\}$	$\{a, b\}$
$\langle a \leftarrow \neg a \cup b \leftarrow b \vee a\rangle(\neg a \wedge b)$	✗	✗	✓	✓
$\langle a \leftarrow \neg a ; b \leftarrow b \vee a\rangle(\neg a \wedge b)$	✗	✗	✗	✓
$\langle (a \leftarrow \neg a ; b \leftarrow b \vee a)^*\rangle(\neg a \wedge b)$	✓	✓	✓	✓

Expressivity of DL-PA: writing programs

- captures the standard programming language primitives:

$$\textbf{skip} \stackrel{\text{def}}{=} \top?$$

$$\textbf{fail} \stackrel{\text{def}}{=} \bot?$$

$$\textbf{if } \varphi \textbf{ then } \pi_1 \textbf{ else } \pi_2 \stackrel{\text{def}}{=} (\varphi?;\pi_1) \cup (\neg\varphi?;\pi_2)$$

$$\textbf{while } \varphi \textbf{ do } \pi \stackrel{\text{def}}{=} (\varphi?;\pi)^*;\neg\varphi?$$

$$\textbf{repeat } \pi \textbf{ until } \varphi \stackrel{\text{def}}{=} \ldots$$

- example: sequential program incrementing an n-bit counter $(p_1 \cdots p_n)$

$$incr(p_1,\ldots,p_n) \stackrel{\text{def}}{=} ;_{i=1\ldots n}\Big(p_i \leftarrow (p_i \leftrightarrow \neg(p_{i-1} \wedge \cdots \wedge p_n))\Big)$$

Outline

Comparing Languages

PROP is *no more expressive* than CNF.
There is a transformation $D : \text{PROP} \to \text{CNF}$ s.t. $\|\varphi\| = \|D(\varphi)\|$
e.g. De Morgan's laws and distributivity

$$D(\varphi_n) = \underbrace{(x_1 \vee \cdots \vee x_n) \wedge (x_1 \vee \cdots \vee x_{n-1} \vee y_n) \wedge \cdots \wedge (y_1 \vee \cdots \vee y_n)}_{2^n \text{ clauses}}$$

PROP-sat is *no more complex* than CNF-sat.
There is a polynomial transformation $T : \text{PROP} \to \text{CNF}$ such that φ is Satisfiable iff $T(\varphi)$ is Satisfiable. e.g. Tseytin transformation

$$T(\varphi_n) = (z_1 \vee \cdots \vee z_n) \wedge (\neg z_1 \vee x_1) \wedge (\neg z_1 \vee y_1) \wedge \cdots \wedge (\neg z_n \vee y_n)$$

PROP is *strictly more succinct* than CNF.
There is no poly transf. $S : \text{PROP} \to \text{CNF}$ s.t. $\|\varphi\| = \|S(\varphi)\|$

Consider $\varphi_n = (x_1 \wedge y_1) \vee (x_2 \wedge y_2) \vee \cdots \vee (x_n \wedge y_n)$

Comparing DLPA and QBF

From QBF to DLPA

- QBF is no more succinct than DLPA: (next slide)
- QBF is no more expressive (follows from succinctness)
- QBF-sat is no more complex (follows from succinctness)

From DLPA to QBF

- DLPA is no more expressive than QBF [2013]
- DLPA-sat is no more complex than QBF-sat
  DLPA-sat $\xrightarrow[\text{[2014]: DLPA-sat in PSPACE}]{}$ PSPACE Turing m. $\xrightarrow[\text{[1973]: QBF is PSPACE-h}]{}$ QBF-sat
- DLPA is no more succinct than QBF [new]

Succinctness: from QBF to DL-PA

Transforming quantifiers

$$\begin{aligned} S(p) &= p \\ S(\varphi_1 \vee \varphi_2) &= S(\varphi_1) \vee S(\varphi_2) \\ S(\neg\varphi) &= \neg S(\varphi) \\ S(\exists P\varphi) &= \langle (p_1{\leftarrow}\top \cup p_1{\leftarrow}\bot); \ldots ; (p_n{\leftarrow}\top \cup p_n{\leftarrow}\bot)\rangle S(\varphi) \end{aligned}$$

where $P = \{p_1, \ldots, p_n\}$

Example:

$$S(\exists a \forall b (b \leftrightarrow c) \vee d) = \langle a{\leftarrow}\top \cup a{\leftarrow}\bot\rangle [b{\leftarrow}\top \cup b{\leftarrow}\bot](b \leftrightarrow c) \vee d$$

Proposition

For any QBF φ, $||S(\varphi)|| = ||\varphi||$ and $S(\varphi)$ is linearly bigger than φ. Therefore, QBF is no more succinct than DLPA.

Succinctness: from DL-PA to QBF

In the other direction, things are not so easy!

A first idea?

$$T(\langle p{\leftarrow}\top\rangle\varphi) = \exists p.(p \wedge T(\varphi))$$
$$T(\langle p{\leftarrow}\bot\rangle\varphi) = \exists p.(\neg p \wedge T(\varphi))$$
$$T(\langle p{\leftarrow}\varphi\rangle\varphi') = \exists p'(p' \leftrightarrow T(\varphi) \wedge T(\varphi'_{p/p'}))$$
$$T(\langle p{\leftarrow}\varphi; p{\leftarrow}\varphi'\rangle\varphi'') = \exists p^1(p^1 \leftrightarrow T(\varphi) \wedge \exists p^2(p^2 \leftrightarrow T(\varphi'_{p/p_1}) \wedge T(\varphi''_{p^2/p})))$$

How will we deal with the Kleene star?

From DLPA to QBF

- idea: $f(\varphi, v) =$ "φ true at v"
- base case: $f(p, v) = p^v$ (propositional var. indexed by valuation name)
- recursive definition of f:

$$f(p, v) = p^v$$
$$f(\neg\varphi, v) = \neg f(\varphi, v)$$
$$f(\varphi_1 \vee \varphi_2, v) = f(\varphi_1, v) \vee f(\varphi_2, v)$$
$$f(\langle\pi\rangle\varphi, v) = \text{“}\exists w\text{”} (g(v, w, \pi) \wedge f(\varphi, w))$$

where:

- "$\exists w$" = $\exists p_1^w \cdots \exists p_n^w$
  - $\{p_1, \cdots, p_n\} = \mathbb{P}_{\varphi_0}$ (variables of φ_0)
- "w is fresh": no p_i^w occurs in φ
- $g(v, w, \pi)$ = "v accesses w via π"

- recursive definition of $g(v, w, \pi)$: ...

From DLPA to QBF

recursive definition of $g(v, w, \pi)$ = "v accesses w via π"

$$g(v, w, p{\leftarrow}\varphi) = (p^w \leftrightarrow f(\varphi, v)) \wedge \bigwedge_{q \in \mathbb{P}_{\varphi_0}, q \neq p} (q^w \leftrightarrow q^v)$$

$$g(v, w, \varphi?) = f(\varphi, v) \wedge \bigwedge_{p \in \mathbb{P}_{\varphi_0}} (p^w \leftrightarrow p^v)$$

$$g(v, w, \pi; \pi') = \text{“}\exists u\text{”} (g(v, u, \pi) \wedge g(u, w, \pi'))$$

$$g(v, w, \pi \cup \pi') = g(v, w, \pi) \vee g(v, w, \pi')$$

⇒ how can we define $g(v, w, \pi^*)$?

$$\pi^* = \textbf{skip} \cup \pi \cup \pi; \pi \cup \ldots \cup \pi^n \cup \ldots$$

$$= \textbf{skip} \cup \pi \cup \pi; \pi \cup \ldots \cup \pi^{2^{|\mathbb{P}_\pi|}}$$

$$= (\textbf{skip} \cup \pi)^{2^{|\mathbb{P}_\pi|}} = (\textbf{skip} \cup \pi); (\textbf{skip} \cup \pi); \ldots; (\textbf{skip} \cup \pi)$$

⇒ how can we define $g(v, w, \pi^*)$ in a non-explosive way?

From DLPA to QBF

- problem: define $g(v, w, \pi^*)$ in a non-explosive way
- solution: divide and conquer [Savitch / Chandra et al., J. / Sipser]
  - $h(v, w, \pi, n)$ = "v accesses w via π in 2^n steps"

$$g(v, w, \pi^*) = h(v, w, \pi, \mathrm{card}(\mathbb{P}_\pi))$$

- recursive definition:

$$h(v, w, \pi, 0) = g(v, w, \pi)$$

$$h(v, w, \pi, n{+}1) = \text{“}\exists u\text{”} \big(h(v, u, \pi, n) \wedge h(u, w, \pi, n)\big) \qquad (\ldots \text{but explosive})$$

$$= \text{“}\exists u\text{”}\ \forall X\ \text{“}\exists v_1\text{”}\ \text{“}\exists w_1\text{”} \Big(h(v_1, w_1, \pi, n) \wedge \big(X \wedge \bigwedge_{p \in \mathbb{P}_{\varphi_0}} ((p^{v_1} \leftrightarrow p^{v}) \wedge (p^{w_1} \leftrightarrow p^{u})) \vee (\neg X \wedge \bigwedge_{p \in \mathbb{P}_{\varphi_0}} ((p^{v_1} \leftrightarrow p^{u}) \wedge (p^{w_1} \leftrightarrow p^{w})))\big)\Big)$$

From DLPA to QBF

Proposition

The length of $f(\varphi_0, v)$ *is quadratic in the length of* φ_0.

Proposition

$v \models \varphi_0$ *iff* $v \models \Big(f(\varphi_0, v)\Big)\Big[\{p^v/p\}_{p \in \mathbb{P}_{\varphi_0}}\Big]$.

Theorem

For every DL-PA *formula there is an equivalent QBF of polynomial length.*

Corollary

DL-PA *and QBF are equally expressive.*
DL-PA *satisfiability checking and* DL-PA *model checking are both PSPACE complete.*

Outline

Decision problems

- Satisfiability: given φ, is $||\varphi|| \neq \emptyset$?
- Validity: given φ, is $||\varphi|| = 2^{\mathbb{P}}$?
- Model Checking: given $V \subseteq \mathbb{P}$ and φ, do we have $V \models \varphi$?

Computational Complexity

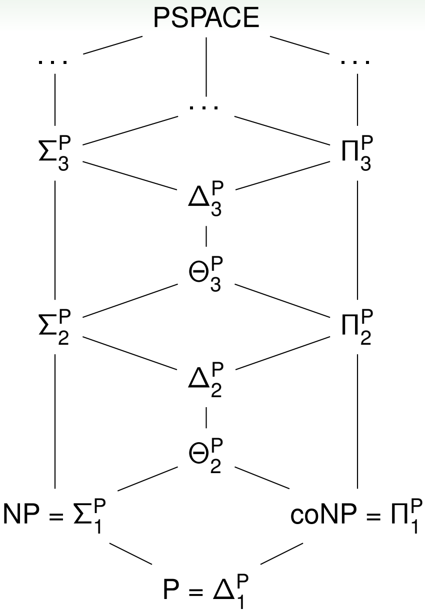
Language	Problem		
	Satisfiability	Validity	Model Checking
PROP	NP-c	coNP-c	in P
CNF	NP-c	in P	in P
DNF	in P	coNP-c	in P

Conj. Normal Form (CNF)

$$\chi ::= p \mid \neg p \mid \chi \vee \chi$$
$$\varphi ::= \chi \mid \varphi \wedge \varphi$$

Disj. Normal Form (DNF)

$$\mu ::= p \mid \neg p \mid \mu \wedge \mu$$
$$\varphi ::= \mu \mid \varphi \vee \varphi$$



A definition

Complexity of the acceptance problem for Turing M. with oracles:

$$\Sigma_0^P = \Pi_0^P = \Delta_0^P = P$$

$$\Sigma_{i+1}^P = NP^{\Sigma_i^P}$$

$$\Pi_{i+1}^P = coNP^{\Sigma_i^P}$$

$$\Delta_{i+1}^P = P^{\Sigma_i^P}$$

$$\Theta_{i+1}^P = L^{\Sigma_i^P}$$

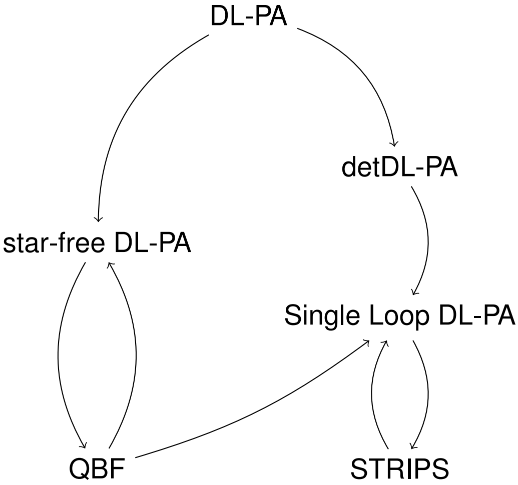
The Polynomial Hierarchy

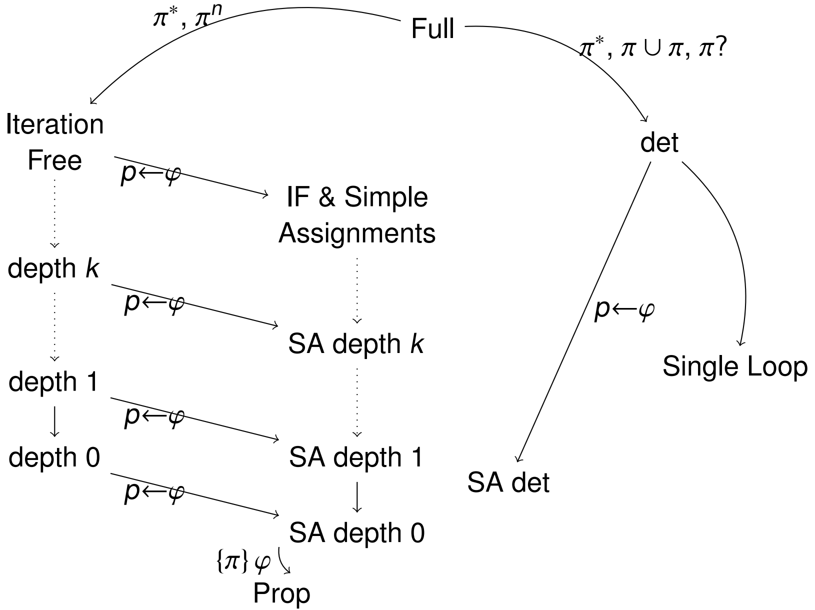
- P: Does this circuit evaluate to true on this given input?
- NP: CIRCUIT Sat, SAT, Sudoku
- coNP: Validity for DNF
- Σ_2^P: Is there a small circuit that does x? Contingent planning.
- PSPACE: Games (generalized tictactoe), QBF, Planning

Deterministic DLPA

$$\begin{array}{llcl}
\mathcal{L}_{\text{fml}} : & \varphi & ::= & p \mid \top \mid \neg\varphi \mid \varphi \vee \varphi \mid \{\pi\}\,\varphi \\
\mathcal{L}_{\text{pgm}} : & \pi & ::= & \textbf{skip} \mid p{\leftarrow}\varphi \mid \pi;\pi \mid \pi^{n}
\end{array}$$

Operator	Depth
$p, \top$	0
$\neg\psi$	$\mu(\psi)$
$\varphi_1 \vee \varphi_2$	$\max\big(\mu(\varphi_1), \mu(\varphi_2)\big)$
$\{\delta\}\,\psi$	$\max\big(\mu(\delta), \mu(\psi)\big)$
$\langle\pi\rangle\psi$	$1 + \max\big(\mu(\pi), \mu(\psi)\big)$
$p{\leftarrow}\varphi$	$\mu(\varphi)$
$\delta_1 ; \delta_2$	$\max\big(\mu(\delta_1), \mu(\delta_2)\big)$
δ^n	∞
δ	$\mu(\delta)$
$\pi_1 ; \pi_2 \quad \pi_1 \cup \pi_2$	$\max\big(\mu(\pi_1), \mu(\pi_2)\big)$
$\varphi?$	$\mu(\varphi)$
π^*	∞
$\exists P\psi$	$1 + \mu(\psi)$


DL-PA
detDL-PA
star-free DL-PA
Single Loop DL-PA
QBF
STRIPS


Full
π^*, π^n
$\pi^*, \pi \cup \pi, \pi?$
Iteration Free
det
$p \leftarrow \varphi$
IF & Simple Assignments
depth k
$p \leftarrow \varphi$
SA depth k
$p \leftarrow \varphi$
Single Loop
depth 1
$p \leftarrow \varphi$
depth 0
SA depth 1
SA det
$p \leftarrow \varphi$
SA depth 0
$\{\pi\}\varphi$
Prop

Bounded alternation logic

	Depth	Satisfiability	Validity	Model Checking
DLPA	i	$\Sigma^{\mathrm{P}}_{i+1}$	Π^{P}_{i+1}	$\Delta^{\mathrm{P}}_{i+1}$
DLPA-SA	i	$\Sigma^{\mathrm{P}}_{i+1}$	Π^{P}_{i+1}	$\Theta^{\mathrm{P}}_{i+1}$
QBF	i	$\Sigma^{\mathrm{P}}_{i+1}$	Π^{P}_{i+1}	$\Theta^{\mathrm{P}}_{i+1}$

Conclusion

DLPA, the language of PSPACE

- Succinctness
- Naturally captures the variety of PSPACE
- Fruitful correspondance with Polynomial Hierarchy

Roadmap

- Domains (From planning, from games)
- Solvers (via QBF + direct)
- Grounding
- . . .