Complexity of Propositional Inclusion and Independence Logic

Jonni Virtema

Leibniz Universität Hannover, Germany
jonni.virtema@gmail.com

Joint work with Miika Hannula, Juha Kontinen, and Heribert Vollmer

Dagstuhl Workshop
25th of June, 2015

Propositional Inclusion and Independence Logic

Grammar of propositional logic $\mathcal{PL}$:

$$\varphi ::= p \mid \neg p \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi).$$

Extensions $\mathcal{PL}$ by inclusion atoms, independence atoms, and classical negation.

$$\varphi ::= p_1, \ldots, p_n \subseteq q_1, \ldots, q_n \mid \vec{r} \perp_{\vec{p}} \vec{q} \mid {\sim}\varphi.$$

The logics are denoted by $\mathcal{PL}[\perp_{\mathrm{c}}, \sim]$, $\mathcal{PL}[\subseteq, \sim]$, etc.

Team Semantics for Propositional Logics

A propositional team is a set of assigments $s : \mathrm{PROP} \to \{0, 1\}$ with the same domain.

Usual team semantics (lax) for atoms and Boolean connectives.

Team Semantics for Propositional Logics

A propositional team is a set of assigments $s : \mathrm{PROP} \to \{0,1\}$ with the same domain.

Usual team semantics (lax) for atoms and Boolean connectives.

$$
\begin{aligned}
X \models {\sim}\varphi \quad &\Leftrightarrow \quad X \not\models \varphi \\
X \models \vec{q} \perp_{\vec{p}} \vec{r} \quad &\Leftrightarrow \quad \forall s, t \in X: \text{ if } s(\vec{p}) = t(\vec{p}) \\
&\qquad \text{then there exists } u \in X : u(\vec{p}\vec{q}) = s(\vec{p}\vec{q}) \text{ and } u(\vec{r}) = t(\vec{r}). \\
X \models \vec{p} \subseteq \vec{q} \quad &\Leftrightarrow \quad \forall s \in X \exists t \in X : s(\vec{p}) = t(\vec{q}).
\end{aligned}
$$

Already $\mathcal{PL}[\sim]$ is highly expressive!

Most connectives studied in team sematics can be defined in $\mathcal{PL}[\sim]$.

The connectives below can be defined in $\mathcal{PL}[\sim]$ with polynomial blow up.

$$
\begin{aligned}
X \models \varphi \mathbin{\char"2A54} \psi \quad &\Leftrightarrow \quad X \models \varphi \text{ or } X \models \psi,\\
X \models \varphi \otimes \psi \quad &\Leftrightarrow \quad \forall Y, Z \subseteq X: \text{ if } Y \cup Z = X, \text{ then } Y \models \varphi \text{ or } Z \models \psi,\\
X \models \varphi \to \psi \quad &\Leftrightarrow \quad \forall Y \subseteq X: \text{if } Y \models \varphi, \text{ then } Y \models \psi,\\
X \models \max(p_1, \dots, p_n) \quad &\Leftrightarrow \quad \{(s(p_1), \dots, s(p_n)) \mid s \in X\} = \{0,1\}^n.
\end{aligned}
$$

Atoms $\subseteq$ and $\perp_c$ can be expressed in $\mathcal{PL}[\sim]$ with exponential blow up.

Already $\mathcal{PL}[\sim]$ is highly expressive!

Most connectives studied in team sematics can be defined in $\mathcal{PL}[\sim]$.

The connectives below can be defined in $\mathcal{PL}[\sim]$ with polynomial blow up.

$$
\begin{aligned}
X \models \varphi \varovee \psi \quad &\Leftrightarrow \quad X \models \varphi \text{ or } X \models \psi,\\
X \models \varphi \otimes \psi \quad &\Leftrightarrow \quad \forall Y, Z \subseteq X: \text{ if } Y \cup Z = X, \text{ then } Y \models \varphi \text{ or } Z \models \psi,\\
X \models \varphi \rightarrow \psi \quad &\Leftrightarrow \quad \forall Y \subseteq X: \text{if } Y \models \varphi, \text{ then } Y \models \psi,\\
X \models \max(p_1, \dots, p_n) \quad &\Leftrightarrow \quad \{(s(p_1), \dots, s(p_n)) \mid s \in X\} = \{0,1\}^n.
\end{aligned}
$$

Atoms $\subseteq$ and $\perp_c$ can be expressed in $\mathcal{PL}[\sim]$ with exponential blow up.

Already $\mathcal{PL}[\sim]$ is highly expressive!

Most connectives studied in team sematics can be defined in $\mathcal{PL}[\sim]$.

The connectives below can be defined in $\mathcal{PL}[\sim]$ with polynomial blow up.

$$
\begin{aligned}
X \models \varphi \ovee \psi \quad &\Leftrightarrow \quad X \models \varphi \text{ or } X \models \psi,\\
X \models \varphi \otimes \psi \quad &\Leftrightarrow \quad \forall Y, Z \subseteq X: \text{ if } Y \cup Z = X, \text{ then } Y \models \varphi \text{ or } Z \models \psi,\\
X \models \varphi \to \psi \quad &\Leftrightarrow \quad \forall Y \subseteq X: \text{if } Y \models \varphi, \text{ then } Y \models \psi,\\
X \models \max(p_1, \dots, p_n) \quad &\Leftrightarrow \quad \{(s(p_1), \dots, s(p_n)) \mid s \in X\} = \{0,1\}^n.
\end{aligned}
$$

Atoms $\subseteq$ and $\perp_{\mathrm{c}}$ can be expressed in $\mathcal{PL}[\sim]$ with exponential blow up.

Expression	Defining $\mathcal{PL}[\sim]$-formula
$\varphi \otimes \psi$	$\sim(\sim\varphi \vee \sim\psi)$
$\varphi \varovee \psi$	$\sim(\sim\varphi \wedge \sim\psi)$
$\varphi \rightarrow \psi$	$(\sim\varphi \varovee \psi) \otimes \sim(p \vee \neg p)$
$\mathrm{dep}(p)$	$p \varovee \neg p$
$\mathrm{dep}(p_1, \ldots, p_n, q)$	$\bigwedge_{i=1}^{n} \mathrm{dep}(p_i) \rightarrow \mathrm{dep}(q)$
$\max(p_1, \ldots, p_n)$	$\sim \bigvee_{i=1}^{n} \mathrm{dep}(p_i)$

PTIME Reductions Between Validity and Satisfiability

Note: $X \models \sim(p \wedge \neg p)$ iff X is non-empty.

For $\varphi \in \mathcal{PL}[\mathcal{C}, \sim]$, define

$$\varphi_{\mathrm{SAT}} := \max(\vec{x}) \rightarrow ((p \vee \neg p) \vee (\varphi \wedge \sim(p \wedge \neg p))),$$
$$\varphi_{\mathrm{VAL}} := \max(\vec{x}) \wedge (\sim(p \wedge \neg p) \rightarrow \varphi),$$

where $\vec{x}$ lists the variables of φ

Theorem

- φ *is satisfiable iff* φ_{SAT} *is valid.*
- φ *is valid iff* φ_{VAL} *is satisfiable.*

PTIME Reductions Between Validity and Satisfiability

Note: $X \models \sim(p \wedge \neg p)$ iff X is non-empty.

For $\varphi \in \mathcal{PL}[\mathcal{C}, \sim]$, define

$$\varphi_{\mathrm{SAT}} := \max(\vec{x}) \to ((p \vee \neg p) \vee (\varphi \wedge \sim(p \wedge \neg p))),$$
$$\varphi_{\mathrm{VAL}} := \max(\vec{x}) \wedge (\sim(p \wedge \neg p) \to \varphi),$$

where $\vec{x}$ lists the variables of φ

Theorem

- *φ is satisfiable iff φ_{SAT} is valid.*
- *φ is valid iff φ_{VAL} is satisfiable.*

PTIME Reductions Between Validity and Satisfiability

Note: $X \models {\sim}(p \wedge \neg p)$ iff X is non-empty.

For $\varphi \in \mathcal{PL}[\mathcal{C}, \sim]$, define

$$\varphi_{\mathrm{SAT}} := \max(\vec{x}) \rightarrow ((p \vee \neg p) \vee (\varphi \wedge {\sim}(p \wedge \neg p))),$$
$$\varphi_{\mathrm{VAL}} := \max(\vec{x}) \wedge ({\sim}(p \wedge \neg p) \rightarrow \varphi),$$

where $\vec{x}$ lists the variables of φ

Theorem

- *φ is satisfiable iff φ_{SAT} is valid.*
- *φ is valid iff φ_{VAL} is satisfiable.*

Complexity Results

Logic	SAT	VAL	MC
$\mathcal{PL}$	NP [0]	coNP [0]	NC_1 [1]
$\mathcal{PL}[\mathrm{dep}(\cdot)]$	NP [3]	NEXPTIME [4]	NP [2]
$\mathcal{PL}[\bot_{\mathrm{c}}]$	NP	in $\mathrm{coNEXPTIME}^{\mathrm{NP}}$	NP
$\mathcal{PL}[\subseteq]$	EXPTIME [5]	coNP	in P [6]
$\mathcal{PL}[\bot_{\mathrm{c}}, \sim]$	AEXPTIME(poly)	AEXPTIME(poly)	PSPACE
$\mathcal{PL}[\subseteq, \sim]$	AEXPTIME(poly)	AEXPTIME(poly)	PSPACE

[0] Cook 1971, Levin 1973, [1] Buss 1987, [2] Ebbing, Lohmann 2012,
[3] Lohmann, Vollmer 2013, [4] V. 2014, [5] Hella, Kuusisto, Meier, Vollmer 2015,
[6] Hella.

Complexity Results

Logic	SAT	VAL	MC
$\mathcal{PL}$	NP[0]	coNP[0]	NC_1[1]
$\mathcal{PL}[\mathrm{dep}(\cdot)]$	NP[3]	NEXPTIME[4]	NP[2]
$\mathcal{PL}[\perp_{\mathrm{c}}]$	NP	in $\mathrm{coNEXPTIME}^{\mathrm{NP}}$	NP
$\mathcal{PL}[\subseteq]$	EXPTIME[5]	coNP	in P[6]
$\mathcal{PL}[\perp_{\mathrm{c}}, \sim]$	AEXPTIME(poly)	AEXPTIME(poly)	PSPACE
$\mathcal{PL}[\subseteq, \sim]$	AEXPTIME(poly)	AEXPTIME(poly)	PSPACE

[0] Cook 1971, Levin 1973, [1] Buss 1987, [2] Ebbing, Lohmann 2012, [3] Lohmann, Vollmer 2013, [4] V. 2014, [5] Hella, Kuusisto, Meier, Vollmer 2015, [6] Hella.

Complexity Results

Logic	SAT	VAL	MC
$\mathcal{PL}$	NP [0]	coNP [0]	NC_1 [1]
$\mathcal{PL}[\mathrm{dep}(\cdot)]$	NP [3]	NEXPTIME [4]	NP [2]
$\mathcal{PL}[\bot_{\mathrm{c}}]$	NP	in $\mathrm{coNEXPTIME}^{\mathrm{NP}}$	NP
$\mathcal{PL}[\subseteq]$	EXPTIME [5]	coNP	in P [6]
$\mathcal{PL}[\bot_{\mathrm{c}}, \sim]$	AEXPTIME(poly)	AEXPTIME(poly)	PSPACE
$\mathcal{PL}[\subseteq, \sim]$	AEXPTIME(poly)	AEXPTIME(poly)	PSPACE

[0] Cook 1971, Levin 1973, [1] Buss 1987, [2] Ebbing, Lohmann 2012,
[3] Lohmann, Vollmer 2013, [4] V. 2014, [5] Hella, Kuusisto, Meier, Vollmer 2015,
[6] Hella.

Complexity Results

Logic	SAT	VAL	MC
$\mathcal{PL}$	NP [0]	coNP [0]	NC_1 [1]
$\mathcal{PL}[\mathrm{dep}(\cdot)]$	NP [3]	NEXPTIME [4]	NP [2]
$\mathcal{PL}[\bot_c]$	NP	in $\mathrm{coNEXPTIME}^{\mathrm{NP}}$	NP
$\mathcal{PL}[\subseteq]$	EXPTIME [5]	coNP	in P [6]
$\mathcal{PL}[\bot_c, \sim]$	AEXPTIME(poly)	AEXPTIME(poly)	PSPACE
$\mathcal{PL}[\subseteq, \sim]$	AEXPTIME(poly)	AEXPTIME(poly)	PSPACE

[0] Cook 1971, Levin 1973, [1] Buss 1987, [2] Ebbing, Lohmann 2012,
[3] Lohmann, Vollmer 2013, [4] V. 2014, [5] Hella, Kuusisto, Meier, Vollmer 2015,
[6] Hella.

Complexity Results

Logic	SAT	VAL	MC
$\mathcal{PL}$	NP [0]	coNP [0]	NC_1 [1]
$\mathcal{PL}[\mathrm{dep}(\cdot)]$	NP [3]	NEXPTIME [4]	NP [2]
$\mathcal{PL}[\bot_{\mathrm{c}}]$	NP	in $\mathrm{coNEXPTIME}^{\mathrm{NP}}$	NP
$\mathcal{PL}[\subseteq]$	EXPTIME [5]	coNP	in P [6]
$\mathcal{PL}[\bot_{\mathrm{c}}, \sim]$	AEXPTIME(poly)	AEXPTIME(poly)	PSPACE
$\mathcal{PL}[\subseteq, \sim]$	AEXPTIME(poly)	AEXPTIME(poly)	PSPACE

[0] Cook 1971, Levin 1973, [1] Buss 1987, [2] Ebbing, Lohmann 2012, [3] Lohmann, Vollmer 2013, [4] V. 2014, [5] Hella, Kuusisto, Meier, Vollmer 2015, [6] Hella.

Idea: SAT for $\mathcal{PL}[\perp_c, \sim]$ is Hard for AEXPTIME(poly)

AEXPTIME(poly) $=$ "alternating exponential time with polynomially many alternations".

We relate AEXPTIME(poly) with alternating polynomial time Turing machines that query to oracles obtained from a quantifier prefix of polynomial length.

Alternation can be replaced by a sequence of word quantifiers

We then relate computations of these deterministic oracle Turing machines with the satisfiability problems of $\mathcal{PL}[\perp_c, \sim]$ and $\mathcal{PL}[\subseteq, \sim]$.

Idea: SAT for $\mathcal{PL}[\perp_c, \sim]$ is Hard for AEXPTIME(poly)

AEXPTIME(poly) = "alternating exponential time with polynomially many alternations".

We relate AEXPTIME(poly) with alternating polynomial time Turing machines that query to oracles obtained from a quantifier prefix of polynomial length.

Alternation can be replaced by a sequence of word quantifiers

We then relate computations of these deterministic oracle Turing machines with the satisfiability problems of $\mathcal{PL}[\perp_c, \sim]$ and $\mathcal{PL}[\subseteq, \sim]$.

Idea: SAT for $\mathcal{PL}[\perp_c, \sim]$ is Hard for AEXPTIME(poly)

AEXPTIME(poly) $=$ "alternating exponential time with polynomially many alternations".

We relate AEXPTIME(poly) with alternating polynomial time Turing machines that query to oracles obtained from a quantifier prefix of polynomial length.

Alternation can be replaced by a sequence of word quantifiers

We then relate computations of these deterministic oracle Turing machines with the satisfiability problems of $\mathcal{PL}[\perp_c, \sim]$ and $\mathcal{PL}[\subseteq, \sim]$.

Characterization via Oracle Machines

The classes Σ_k^{EXP} and Π_k^{EXP} of the exponential time hierarchy are characterized by polynomial-time constant-alternation oracle Turing machines that query to k oracles (Orponen 1983).

Theorem

A set A belongs to the class AEXPTIME(poly) iff there exist a polynomial f and a polynomial-time alternating oracle Turing machine M such that, for all x,

$$x \in A \textit{ iff } Q_1 A_1 \ldots Q_{f(n)} A_{f(n)} (M \textit{ accepts } x \textit{ with oracles } (A_1, \ldots, A_{f(n)})),$$

where n is the length of x and $Q_1, \ldots, Q_{f(n)}$ alternate between $\exists$ and $\forall$, i.e $Q_{i+1} \in \{\forall, \exists\} \setminus \{Q_i\}$.

Characterization Without Alternation

Alternating Turing machine can be replaced by a sequence of word quantifiers over a deterministic Turing machine (Chandra, Kozen, and Stockmeyer 1981).

Theorem

A set A belongs to the class AEXPTIME(poly) iff there exists a polynomial-time deterministic oracle Turing machine M^* such that $x \in A$ iff

$$Q_1 A_1 \dots Q_{f(n)} A_{f(n)} Q'_1 \vec{y}_1 \dots Q'_{g(n)} \vec{y}_{g(n)}$$
$$(M^* \textit{ accepts } (x, \vec{y}_1, \dots, \vec{y}_{g(n)}) \textit{ with oracle } (A_1, \dots, A_{f(n)})),$$

where $Q_1, \dots, Q_{f(n)}$ and $Q'_1, \dots, Q'_{g(n)}$ are alternating sequences of quantifiers $\exists$ and $\forall$, and each $\vec{y}_i$ is a $g(n)$-ary sequence of propositional variables where n is the length of x.

g is a polynomial that bounds the running time of M.

From Turing Machines to $\mathrm{SAT}(\mathcal{PL}[\subseteq, \sim])$

The whole computation of an oracle Turing machine is encoded to a team X.

Encoded information is accessed via expressions of the form:

$$\exists s \in X \text{ s.t. } \{s\} \models \varphi, \text{ where } \varphi \text{ is in } \mathcal{PL}.$$

In $\mathcal{PL}[\sim]$ the above is written as $X \models \sim\neg\varphi$.

Example

The membership of a binary string $\vec{a}$ in an oracle A_i is expressed by

$$X \models \sim\neg(\vec{q} = \vec{a} \wedge \vec{r} = \mathrm{bin}(i)).$$

Tuple $\vec{q}$ lists the propositions used to encode the content of oracles.

Tuple $\vec{r}$ encodes the indices of the oracles.

Simulating Quantification

Recall:

- The whole computation is encoded in a team.
- Idea of encoding: $\exists s \in X$ s.t. $\{s\} \models \varphi$.
- $X \models \varphi \otimes \psi$ iff $\forall Y, Z$ s.t. $Y \cup Z = X$: $Y \models \varphi$ or $Z \models \psi$.
- $X \models \varphi \vee \psi$ iff $\exists Y, Z$ s.t. $Y \cup Z = X$: $Y \models \varphi$ and $Z \models \psi$.

We use $\vee$ to simulate existential quantification of relations and points.

We use $\otimes$ to simulate universal quantification of relations and points.

Simulating Quantification

Recall:

- The whole computation is encoded in a team.
- Idea of encoding: $\exists s \in X$ s.t. $\{s\} \models \varphi$.
- $X \models \varphi \otimes \psi$ iff $\forall Y, Z$ s.t. $Y \cup Z = X$: $Y \models \varphi$ or $Z \models \psi$.
- $X \models \varphi \vee \psi$ iff $\exists Y, Z$ s.t. $Y \cup Z = X$: $Y \models \varphi$ and $Z \models \psi$.

We use $\vee$ to simulate existential quantification of relations and points.

We use $\otimes$ to simulate universal quantification of relations and points.

Idea of Quantification

- Fix the domain D of the encoding.
- $\exists Y \varphi(Y) \quad \mapsto \quad \exists A \subseteq D \colon \varphi(D \setminus A)$.
- $\forall Y \varphi(Y) \quad \mapsto \quad \forall A \subseteq D \colon \varphi(D \setminus A)$.
- Maintain uniformity in quantification. (Arities of A and D do not coincide.)

Example of Quantification

Our encoding uses variables $p_1, \ldots, p_n$: $\max(p_1, \ldots, p_n)$

Existential quantification of the oracle A_i: $\vec{r} = \mathrm{bin}(i) \vee (\alpha \wedge \varphi)$.

Formula α takes care of the uniformity. ($\subseteq$ or $\perp_{\mathrm{c}}$ needed)

$\alpha := \max(\vec{y}) \wedge \vec{y} \perp \vec{q}\vec{r}$

r encodes names of oracles, q encodes content of oracles, y encodes everything else.

Example of Quantification

Our encoding uses variables $p_1, \ldots, p_n$: $\max(p_1, \ldots, p_n)$

Existential quantification of the oracle A_i: $\vec{r} = \mathrm{bin}(i) \vee (\alpha \wedge \varphi)$.

Formula α takes care of the uniformity. ($\subseteq$ or $\perp_{\mathrm{c}}$ needed)

$\alpha := \max(\vec{y}) \wedge \vec{y} \perp \vec{q}\vec{r}$

r encodes names of oracles, q encodes content of oracles, y encodes everything else.

Example of Quantification

Our encoding uses variables $p_1, \ldots, p_n$: $\max(p_1, \ldots, p_n)$

Existential quantification of the oracle A_i: $\vec{r} = \mathrm{bin}(i) \vee (\alpha \wedge \varphi)$.

Formula α takes care of the uniformity. ($\subseteq$ or $\perp_{\mathrm{c}}$ needed)

$\alpha := \max(\vec{y}) \wedge \vec{y} \perp \vec{q}\vec{r}$

r encodes names of oracles, q encodes content of oracles, y encodes everything else.

Example of Quantification

Our encoding uses variables $p_1, \ldots, p_n$: $\max(p_1, \ldots, p_n)$

Existential quantification of the oracle A_i: $\vec{r} = \mathrm{bin}(i) \vee (\alpha \wedge \varphi)$.

Formula α takes care of the uniformity. ($\subseteq$ or $\perp_{\mathrm{c}}$ needed)

$\alpha := \max(\vec{y}) \wedge \vec{y} \perp \vec{q}\vec{r}$

r encodes names of oracles, q encodes content of oracles, y encodes everything else.

Complexity of $\mathcal{PL}[\perp_c, \sim]$

Theorem

$\mathrm{SAT}(\mathcal{PL}[\perp_c, \sim])$ *is* AEXPTIME(poly)*-complete.*

Proof.

Hardness: Done.
Membership: Guess a possibly exponential-size team T and do APTIME model checking. □

Corollary

$\mathrm{VAL}(\mathcal{PL}[\perp_c, \sim])$ *is* AEXPTIME(poly)*-complete.*

Complexity of $\mathcal{PL}[\perp_c, \sim]$

Theorem

$\mathrm{SAT}(\mathcal{PL}[\perp_c, \sim])$ *is* AEXPTIME(poly)*-complete.*

Proof.

Hardness: Done.
Membership: Guess a possibly exponential-size team T and do APTIME model checking. □

Corollary

$\mathrm{VAL}(\mathcal{PL}[\perp_c, \sim])$ *is* AEXPTIME(poly)*-complete.*

Further Complexity Results

Theorem

$\mathrm{SAT}(\mathcal{PL}[\subseteq, \sim])$ *and* $\mathrm{VAL}(\mathcal{PL}[\subseteq, \sim])$ *are* AEXPTIME(poly)*-complete.*

Theorem

$\mathrm{MC}(\mathcal{PL}[\subseteq, \sim])$ *and* $\mathrm{MC}(\mathcal{PL}[\perp_c, \sim])$ *are* PSPACE*-complete*

Further Complexity Results

Theorem

$\mathrm{SAT}(\mathcal{PL}[\subseteq, \sim])$ *and* $\mathrm{VAL}(\mathcal{PL}[\subseteq, \sim])$ *are* AEXPTIME(poly)*-complete.*

Theorem

$\mathrm{MC}(\mathcal{PL}[\subseteq, \sim])$ *and* $\mathrm{MC}(\mathcal{PL}[\perp_{\mathrm{c}}, \sim])$ *are* PSPACE*-complete*

Further Complexity Results

Theorem

$\mathrm{VAL}(\mathcal{PL}[\subseteq])$ *is* coNP*-complete*

Proof.

Hardness: $\mathrm{VAL}(\mathcal{PL})$ is coNP-complete.
Membership:

1. $\mathcal{PL}[\subseteq]$ is union closed.
2. $\varphi \in \mathcal{PL}[\subseteq]$ is valid iff φ is valid on singleton teams.
3. On singleton teams inclusion atoms can be eliminated.
4. Check validity of the $\mathcal{PL}$-translatee.

$\square$

Complexity Results

Logic	SAT	VAL	MC
$\mathcal{PL}$	NP [0]	coNP [0]	NC_1 [1]
$\mathcal{PL}[\mathrm{dep}(\cdot)]$	NP [3]	NEXPTIME [4]	NP [2]
$\mathcal{PL}[\bot_c]$	NP	in $\mathrm{coNEXPTIME}^{\mathrm{NP}}$	NP
$\mathcal{PL}[\subseteq]$	EXPTIME [5]	coNP	in P [6]
$\mathcal{PL}[\bot_c, \sim]$	AEXPTIME(poly)	AEXPTIME(poly)	PSPACE
$\mathcal{PL}[\subseteq, \sim]$	AEXPTIME(poly)	AEXPTIME(poly)	PSPACE

[0] Cook 1971, Levin 1973, [1] Buss 1987, [2] Ebbing, Lohmann 2012,
[3] Lohmann, Vollmer 2013, [4] V. 2014, [5] Hella, Kuusisto, Meier, Vollmer 2015,
[6] Hella.

Complexity Results

Thanks!

Logic	SAT	VAL	MC
$\mathcal{PL}$	NP [0]	coNP [0]	NC_1 [1]
$\mathcal{PL}[\mathrm{dep}(\cdot)]$	NP [3]	NEXPTIME [4]	NP [2]
$\mathcal{PL}[\perp_c]$	NP	in $\mathrm{coNEXPTIME}^{\mathrm{NP}}$	NP
$\mathcal{PL}[\subseteq]$	EXPTIME [5]	coNP	in P [6]
$\mathcal{PL}[\perp_c, \sim]$	AEXPTIME(poly)	AEXPTIME(poly)	PSPACE
$\mathcal{PL}[\subseteq, \sim]$	AEXPTIME(poly)	AEXPTIME(poly)	PSPACE

[0] Cook 1971, Levin 1973, [1] Buss 1987, [2] Ebbing, Lohmann 2012,
[3] Lohmann, Vollmer 2013, [4] V. 2014, [5] Hella, Kuusisto, Meier, Vollmer 2015,
[6] Hella.