Model checking and validity in propositional and modal inclusion logics

Jonni Virtema

Hasselt University, Belgium
jonni.virtema@gmail.com

Joint work with Lauri Hella[1], Antti Kuusisto[2], and Arne Meier[3]

[1]University of Tampere, Finland, [2]University of Bremen, Germany, [3]University of Hanover, Germany

23rd of August, 2017 – MFCS 2017

Core of Team Semantics

- In most studied logics formulae are evaluated in a single state of affairs. E.g.,
  - a first-order assignment in first-order logic,
  - a propositional assignment in propositional logic,
  - a possible world of a Kripke structure in modal logic.
- In team semantics sets of states of affairs are considered. E.g.,
  - a set of first-order assignments in first-order logic,
  - a set of propositional assignments in propositional logic,
  - a set of possible worlds of a Kripke structure in modal logic.
- These sets of things are called teams.

Core of Team Semantics

- In most studied logics formulae are evaluated in a single state of affairs. E.g.,
  - a first-order assignment in first-order logic,
  - a propositional assignment in propositional logic,
  - a possible world of a Kripke structure in modal logic.
- In team semantics sets of states of affairs are considered. E.g.,
  - a set of first-order assignments in first-order logic,
  - a set of propositional assignments in propositional logic,
  - a set of possible worlds of a Kripke structure in modal logic.
- These sets of things are called teams.

Team Semantics: Motivation and History

Logical modelling of uncertainty, imperfect information, and different notions of dependence such as functional dependence and independence. Related to similar concepts in statistics, database theory etc.

Historical development:

- Branching quantifiers by Henkin 1959.
  $\begin{pmatrix} \forall x \exists y \\ \forall x' \exists y' \end{pmatrix} \varphi(x, y, x', y')$
- Independence-friendly logic by Hintikka and Sandu 1989.
  $\forall x \exists y \forall x' \exists y' / \{x, y\}\, \varphi(x, y, x', y')$
- Compositional semantics for independence-friendly logic by Hodges 1997. (Origin of team semantics.)
- Dependence logic and modal dependence logic by Väänänen 2007.
- Introduction of other dependency notions to team semantics such as inclusion, exclusion, and independence. Galliani, Grädel, Väänänen.
- Generalised atoms by Kuusisto (derived from generalised quantifiers).
- Multiteam and polyteam semantics by Hannula et al.

Team Semantics: Motivation and History

Logical modelling of uncertainty, imperfect information, and different notions of dependence such as functional dependence and independence. Related to similar concepts in statistics, database theory etc.

Historical development:

- Branching quantifiers by Henkin 1959.
- Independence-friendly logic by Hintikka and Sandu 1989.
- Compositional semantics for independence-friendly logic by Hodges 1997. (Origin of team semantics.)
- Dependence logic and modal dependence logic by Väänänen 2007.
- Introduction of other dependency notions to team semantics such as inclusion, exclusion, and independence. Galliani, Grädel, Väänänen.
- Generalised atoms by Kuusisto (derived from generalised quantifiers).
- Multiteam and polyteam semantics by Hannula et al.

Inclusion logics in first-order setting

We study logics with inclusion dependencies:
For a set of first-order assignments X

$$X \models \vec{x} \subseteq \vec{y} \quad \text{iff} \quad \forall s \in X \exists s' \in X : s(\vec{x}) = s'(\vec{y}).$$

In first-order setting $\mathrm{FO}(\subseteq)$ has very interesting properties:

- $\mathrm{FO}(\subseteq)$ has the same expressive power as posGFP.
- $\mathrm{FO}(\subseteq)$ with strict semantics has the same expressive power as ESO.
- Fragments of $\mathrm{FO}(\subseteq)$ with strict semantics capture $\mathrm{NTIME}_{\mathrm{RAM}}(n^k)$, fixed k.

Inclusion logics in first-order setting

We study logics with inclusion dependencies:
For a set of first-order assignments X

$$X \models \vec{x} \subseteq \vec{y} \quad \text{iff} \quad \forall s \in X \exists s' \in X : s(\vec{x}) = s'(\vec{y}).$$

In first-order setting $\mathrm{FO}(\subseteq)$ has very interesting properties:

- $\mathrm{FO}(\subseteq)$ has the same expressive power as posGFP.
- $\mathrm{FO}(\subseteq)$ with strict semantics has the same expressive power as ESO.
- Fragments of $\mathrm{FO}(\subseteq)$ with strict semantics capture $\mathrm{NTIME}_{\mathrm{RAM}}(n^k)$, fixed k.

Inclusion logics in propositional setting

For a set of propositional assignments X and $\vec{\varphi}, \vec{\psi} \in \mathrm{PL}$

$$X \models \vec{\varphi} \subseteq \vec{\psi} \quad \text{iff} \quad \forall s \in X \exists s' \in X : s(\vec{\varphi}) = s'(\vec{\psi}).$$

In propositional setting $\mathrm{PL}(\subseteq)$ and $\mathrm{ML}(\subseteq)$ have interesting properties:

- $\mathrm{PL}(\subseteq)$ definable classes of propositional teams are exactly those $\mathcal{C}$ s.t.
  - $\emptyset \in \mathcal{C}$ and
  - $\mathcal{C}$ is union closed ($X \in \mathcal{C}, Y \in \mathcal{C} \Rightarrow X \cup Y \in \mathcal{C}$).
- $\mathrm{ML}(\subseteq)$ definable classes of Kripke models with teams are those $\mathcal{C}$ s.t.
  - $(\mathrm{K}, \emptyset) \in \mathcal{C}$, for every K,
  - $\mathcal{C}$ is union closed ($(\mathrm{K}, X) \in \mathcal{C}, (\mathrm{K}, Y) \in \mathcal{C} \Rightarrow (\mathrm{K}, X \cup Y) \in \mathcal{C}$),
  - $\mathcal{C}$ is closed under team k-bisimulation for some k.

Inclusion logics in propositional setting

For a set of propositional assignments X and $\vec{\varphi}, \vec{\psi} \in \mathrm{PL}$

$$X \models \vec{\varphi} \subseteq \vec{\psi} \quad \text{iff} \quad \forall s \in X \exists s' \in X : s(\vec{\varphi}) = s'(\vec{\psi}).$$

In propositional setting $\mathrm{PL}(\subseteq)$ and $\mathrm{ML}(\subseteq)$ have interesting properties:

- $\mathrm{PL}(\subseteq)$ definable classes of propositional teams are exactly those $\mathcal{C}$ s.t.
  - $\emptyset \in \mathcal{C}$ and
  - $\mathcal{C}$ is union closed ($X \in \mathcal{C}, Y \in \mathcal{C} \Rightarrow X \cup Y \in \mathcal{C}$).
- $\mathrm{ML}(\subseteq)$ definable classes of Kripke models with teams are those $\mathcal{C}$ s.t.
  - $(\mathrm{K}, \emptyset) \in \mathcal{C}$, for every K,
  - $\mathcal{C}$ is union closed ($(\mathrm{K}, X) \in \mathcal{C}, (\mathrm{K}, Y) \in \mathcal{C} \Rightarrow (\mathrm{K}, X \cup Y) \in \mathcal{C}$),
  - $\mathcal{C}$ is closed under team k-bisimulation for some k.

Inclusion logics in propositional setting

For a set of propositional assignments X and $\vec{\varphi}, \vec{\psi} \in \mathrm{PL}$

$$X \models \vec{\varphi} \subseteq \vec{\psi} \quad \text{iff} \quad \forall s \in X \exists s' \in X : s(\vec{\varphi}) = s'(\vec{\psi}).$$

In propositional setting $\mathrm{PL}(\subseteq)$ and $\mathrm{ML}(\subseteq)$ have interesting properties:

- $\mathrm{PL}(\subseteq)$ definable classes of propositional teams are exactly those $\mathcal{C}$ s.t.
  - $\emptyset \in \mathcal{C}$ and
  - $\mathcal{C}$ is union closed ($X \in \mathcal{C}, Y \in \mathcal{C} \Rightarrow X \cup Y \in \mathcal{C}$).
- $\mathrm{ML}(\subseteq)$ definable classes of Kripke models with teams are those $\mathcal{C}$ s.t.
  - $(\mathrm{K}, \emptyset) \in \mathcal{C}$, for every K,
  - $\mathcal{C}$ is union closed ($(\mathrm{K}, X) \in \mathcal{C}, (\mathrm{K}, Y) \in \mathcal{C} \Rightarrow (\mathrm{K}, X \cup Y) \in \mathcal{C}$),
  - $\mathcal{C}$ is closed under team k-bisimulation for some k.

Propositional team semantics

Syntax of propositional logic:

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

Semantics via propositional assignments:

	p	q	r
s	0	1	1

$$s \models q \wedge r$$

Team semantics / semantics via sets of assignments:

	p	q	r
s	0	1	1
t	1	1	0
u	0	1	0

$$\{s, t, u\} \models q, \quad \{s, t\} \models p \vee r$$

Team semantics

We want that for each formula φ of propositional logic and for each team X

$$X \models \varphi \quad \text{iff} \quad \forall s \in X : s \models \varphi.$$

Team semantics

We want that for each formula φ of propositional logic and for each team X

$$X \models \varphi \quad \text{iff} \quad \forall s \in X : s \models \varphi.$$

We define that

$$\begin{aligned}
X \models p \quad & \text{iff} \quad \forall s \in X : s(p) = 1 \\
X \models \neg p \quad & \text{iff} \quad \forall s \in X : s(p) = 0 \\
X \models \varphi \wedge \psi \quad & \text{iff} \quad X \models \varphi \text{ and } X \models \psi \\
X \models \varphi \vee \psi \quad & \text{iff} \quad Y \models \varphi \text{ and } Z \models \psi, \\
& \phantom{\text{iff}} \quad \text{for some } Y, Z \subseteq X \text{ such that } Y \cup Z = X.
\end{aligned}$$

Strict team semantics

We want that for each formula φ of propositional logic and for each team X

$$X \models_s \varphi \quad \text{iff} \quad \forall s \in X : s \models \varphi.$$

We define that

$$\begin{aligned}
X \models_s p \quad & \text{iff} \quad \forall s \in X : s(p) = 1 \\
X \models_s \neg p \quad & \text{iff} \quad \forall s \in X : s(p) = 0 \\
X \models_s \varphi \wedge \psi \quad & \text{iff} \quad X \models_s \varphi \text{ and } X \models_s \psi \\
X \models_s \varphi \vee \psi \quad & \text{iff} \quad Y \models_s \varphi \text{ and } Z \models_s \psi, \\
& \quad \text{for some } Y, Z \subseteq X \text{ such that } Y \cup Z = X \\
& \quad \text{and } Y \cap Z = \emptyset.
\end{aligned}$$

Propositional inclusion logic

We extend PL by inclusion atoms: $(p_1, p_2) \subseteq (q_1, q_2)$

"truth values that appear for p_1, p_2 also appear as truth values for q_1, q_2".

	p	q	r
s	1	0	0
t	1	1	1
u	0	1	0

$\{s,t\} \not\models (p,q) \subseteq (q,r), \quad \{s,t\} \models (p,p) \subseteq (q,r)$

Define $\varphi := (p \wedge (p \subseteq r)) \vee (q \wedge (q \subseteq r))$. Now

$$\{s,t,u\} \models \varphi, \text{ but } \{s,t,u\} \not\models_s \varphi.$$

Propositional inclusion logic

We extend PL by inclusion atoms: $(p_1, p_2) \subseteq (q_1, q_2)$

"truth values that appear for p_1, p_2 also appear as truth values for q_1, q_2".

	p	q	r
s	1	0	0
t	1	1	1
u	0	1	0

$\{s,t\} \not\models (p,q) \subseteq (q,r), \quad \{s,t\} \models (p,p) \subseteq (q,r)$

Define $\varphi := (p \wedge (p \subseteq r)) \vee (q \wedge (q \subseteq r))$. Now

$$\{s,t,u\} \models \varphi, \text{ but } \{s,t,u\} \not\models_s \varphi.$$

Important decision problems

Model checking:
Input: A team X and a formula φ.
Output: Does $X \models \varphi$ hold?

Satisfiability:
Input: A formula φ.
Output: Does there exists a non-empty team X s.t. $X \models \varphi$?

Validity:
Input: A formula φ.
Output: Does $X \models \varphi$ hold for every non-empty team X?

Complexity results

	Satisfiability		Validity		Model checking	
	strict	lax	strict	lax	strict	lax
PL	— NP [Cook 71, Levin 73] —		—— coNP [Cook 71, Levin 73] ——		— NC^1 [Buss 87] —	
PL(⊆)	EXPTIME [†]	EXPTIME [†]	coNP	coNP [‡]	NP	P
ML	—— PSPACE [Ladner 77] ——		——— PSPACE [Ladner 77] ———		P [Clarke et al. 86]	
ML(⊆)	EXPTIME [†]	EXPTIME [†]	coNEXPTIME-h	coNEXPTIME-h	NP	P

† Hella et al. 2015, ‡ Hannula et al. 2015

Complexity results

	Satisfiability		Validity		Model checking	
	strict	lax	strict	lax	strict	lax
PL	— NP [Cook 71, Levin 73] —		—— coNP [Cook 71, Levin 73] ——		— NC^1 [Buss 87] —	
PL(⊆)	EXPTIME [†]	EXPTIME [†]	coNP	coNP [‡]	NP	P
ML	—— PSPACE [Ladner 77] ——		——— PSPACE [Ladner 77] ———		P [Clarke et al. 86]	
ML(⊆)	EXPTIME [†]	EXPTIME [†]	coNEXPTIME-h	coNEXPTIME-h	NP	P

† Hella et al. 2015, ‡ Hannula et al. 2015

Proof techniques

- $\mathrm{MC}(\mathsf{PL}(\subseteq))$ is P-hard: Reduction from the monotone circuit value problem.
- $\mathrm{MC}(\mathsf{ML}(\subseteq)) \in$ P: Recursive monotone labelling algorithm.

- $\mathrm{MC}(\mathsf{PL}_s(\subseteq))$ is NP-hard: Reduction from the set splitting problem.
- $\mathrm{MC}(\mathsf{ML}_s(\subseteq)) \in$ NP: Brute force algorithm.

- $\mathrm{VAL}(\mathsf{ML}(\subseteq))$ is coNEXPTIME-hard: Reduction from the complement of DQBF.

Proof techniques

- $\mathrm{MC}(\mathsf{PL}(\subseteq))$ is P-hard: Reduction from the monotone circuit value problem.
- $\mathrm{MC}(\mathsf{ML}(\subseteq)) \in$ P: Recursive monotone labelling algorithm.

- $\mathrm{MC}(\mathsf{PL}_s(\subseteq))$ is NP-hard: Reduction from the set splitting problem.
- $\mathrm{MC}(\mathsf{ML}_s(\subseteq)) \in$ NP: Brute force algorithm.

- $\mathrm{VAL}(\mathsf{ML}(\subseteq))$ is coNEXPTIME-hard: Reduction from the complement of DQBF.

Proof techniques

- $\mathrm{MC}(\mathsf{PL}(\subseteq))$ is P-hard: Reduction from the monotone circuit value problem.
- $\mathrm{MC}(\mathsf{ML}(\subseteq)) \in \mathsf{P}$: Recursive monotone labelling algorithm.

- $\mathrm{MC}(\mathsf{PL}_s(\subseteq))$ is NP-hard: Reduction from the set splitting problem.
- $\mathrm{MC}(\mathsf{ML}_s(\subseteq)) \in \mathsf{NP}$: Brute force algorithm.

- $\mathrm{VAL}(\mathsf{ML}(\subseteq))$ is coNEXPTIME-hard: Reduction from the complement of DQBF.

Monotone circuit value problem

Monotone circuit is a finite directed, acyclic graph in which each node is either:

- an input gate labelled with a Boolean variable x_i,
- a disjunction gate with indegree 2,
- a conjunction gate with indegree 2.

There is exactly one node with outdegree 0, called the output gate.

Decision problem:

Input: Monotone circuit $\mathcal{C}$ and values for the Boolean variables x_i.

Output: Is the value of the output gate 1?

Monotone circuit value problem is P-complete.

Monotone circuit value problem

Monotone circuit is a finite directed, acyclic graph in which each node is either:

- an input gate labelled with a Boolean variable x_i,
- a disjunction gate with indegree 2,
- a conjunction gate with indegree 2.

There is exactly one node with outdegree 0, called the output gate.

Decision problem:

Input: Monotone circuit $\mathcal{C}$ and values for the Boolean variables x_i.

Output: Is the value of the output gate 1?

Monotone circuit value problem is P-complete.

Idea of the reduction

- Consider each gate s_i as an assignment s.t. $s_i(p_i) = 1$ and $s_i(p_j) = 0$.
- If s_k is a disjunction gate of s_i and s_j then $s_i(p_{k=i \vee j}) = 1$.

After skipping some technicalities we have that

$$
\begin{aligned}
X \models \top \subseteq p_0 \text{ iff } \quad & s_0 \in X \\
X \models p_i \subseteq p_j \text{ iff } \quad & s_i \in X \text{ implies } s_j \in X \\
X \models p_k \subseteq p_{k=i \vee j} \text{ iff } \quad & s_k \in X \text{ implies that } s_i \in X \text{ or } s_j \in X
\end{aligned}
$$

The idea is that gates that are in the team X have a value 1.

Let X be the set of Boolean gates and those input gates that get the input 1. Now

$$X \models \neg p_\perp \vee (\psi_{\mathrm{out}=1} \wedge \psi_\wedge \wedge \psi_\vee) \quad \text{iff} \quad \text{output of the circuit is } 1.$$

Idea of the reduction

- Consider each gate s_i as an assignment s.t. $s_i(p_i) = 1$ and $s_i(p_j) = 0$.
- If s_k is a disjunction gate of s_i and s_j then $s_i(p_{k=i\vee j}) = 1$.

After skipping some technicalities we have that

$$\begin{aligned} X \models \top \subseteq p_0 \text{ iff } \quad & s_0 \in X \\ X \models p_i \subseteq p_j \text{ iff } \quad & s_i \in X \text{ implies } s_j \in X \\ X \models p_k \subseteq p_{k=i\vee j} \text{ iff } \quad & s_k \in X \text{ implies that } s_i \in X \text{ or } s_j \in X \end{aligned}$$

The idea is that gates that are in the team X have a value 1.

Let X be the set of Boolean gates and those input gates that get the input 1. Now

$$X \models \neg p_\perp \vee (\psi_{\mathrm{out}=1} \wedge \psi_\wedge \wedge \psi_\vee) \quad \text{iff} \quad \text{output of the circuit is } 1.$$

Idea of the reduction

- Consider each gate s_i as an assignment s.t. $s_i(p_i) = 1$ and $s_i(p_j) = 0$.
- If s_k is a disjunction gate of s_i and s_j then $s_i(p_{k=i\vee j}) = 1$.

After skipping some technicalities we have that

$$X \models \top \subseteq p_0 \text{ iff } \quad s_0 \in X$$
$$X \models p_i \subseteq p_j \text{ iff } \quad s_i \in X \text{ implies } s_j \in X$$
$$X \models p_k \subseteq p_{k=i\vee j} \text{ iff } \quad s_k \in X \text{ implies that } s_i \in X \text{ or } s_j \in X$$

The idea is that gates that are in the team X have a value 1.

Let X be the set of Boolean gates and those input gates that get the input 1. Now

$$X \models \neg p_\bot \vee (\psi_{\mathrm{out}=1} \wedge \psi_\wedge \wedge \psi_\vee) \quad \text{iff} \quad \text{output of the circuit is 1.}$$

P algorithm for $\mathrm{MC}(\mathsf{PL}(\subseteq))$ and $\mathrm{MC}(\mathsf{ML}(\subseteq))$

Important properties:

- Each team X has a unique maximal subteam satisfying a given formula φ.
- For literals $\mathrm{maxsub}(X, \varphi)$ is computable in polynomial time.

Idea of the algorithm checking whether $X \models \varphi$:

1. Build the syntactic tree of φ and label each of its nodes with X.
2. Bottom up part of the algorithm:
   2.1 For literals φ labelled by Y, replace Y by $\mathrm{maxsub}(Y, \varphi)$.
   2.2 For other nodes; update their label depending on their connective, their previous label and their child nodes new labels.
3. Top down part of the algorithm:
   3.1 Starting from root, update labels depending on the connective, previous label and the parent nodes new label.
4. Go to 2.

The labelling algorithm is decreasing and each round takes only polynomial time.

P algorithm for $\mathrm{MC}(\mathsf{PL}(\subseteq))$ and $\mathrm{MC}(\mathsf{ML}(\subseteq))$

Important properties:

- Each team X has a unique maximal subteam satisfying a given formula φ.
- For literals $\mathrm{maxsub}(X, \varphi)$ is computable in polynomial time.

Idea of the algorithm checking whether $X \models \varphi$:

1. Build the syntactic tree of φ and label each of its nodes with X.
2. Bottom up part of the algorithm:
   2.1 For literals φ labelled by Y, replace Y by $\mathrm{maxsub}(Y, \varphi)$.
   2.2 For other nodes; update their label depending on their connective, their previous label and their child nodes new labels.
3. Top down part of the algorithm:
   3.1 Starting from root, update labels depending on the connective, previous label and the parent nodes new label.
4. Go to 2.

The labelling algorithm is decreasing and each round takes only polynomial time.

P algorithm for $\mathrm{MC}(\mathsf{PL}(\subseteq))$ and $\mathrm{MC}(\mathsf{ML}(\subseteq))$

Important properties:

- Each team X has a unique maximal subteam satisfying a given formula φ.
- For literals $\mathrm{maxsub}(X, \varphi)$ is computable in polynomial time.

Idea of the algorithm checking whether $X \models \varphi$:

1. Build the syntactic tree of φ and label each of its nodes with X.
2. Bottom up part of the algorithm:
   2.1 For literals φ labelled by Y, replace Y by $\mathrm{maxsub}(Y, \varphi)$.
   2.2 For other nodes; update their label depending on their connective, their previous label and their child nodes new labels.
3. Top down part of the algorithm:
   3.1 Starting from root, update labels depending on the connective, previous label and the parent nodes new label.
4. Go to 2.

The labelling algorithm is decreasing and each round takes only polynomial time.

Set splitting problem

Set splitting problem is the following decision problem:

Input: A finite family $\mathcal{F} = \{S_1, \ldots, S_n\}$ of subsets of a finite set S.

Output: Do there exist subsets $L \subseteq S$ and $R \subseteq S$ such that:

- L and R is a partition of S,
- for each $S_i \in \mathcal{F}$ there exists $a, b \in S_i$ s.t. $a \in L$ and $b \in R$.

Set splitting problem is NP-complete.

Idea of the reduction from set splitting

Let $\mathcal{F} = \{Q_1, \ldots Q_n\}$, $S = \{s_i, \ldots, s_k\}$ be an instance of the problem.

- Consider each point s_i as an assignment s.t. $s_i(p_i) = 1$ and $s_i(p_j) = 0$.
- Sets Q_j are encoded s.t. $s_i(q_j) = 1$ iff $s_i \in Q_j$.

Define $X := \{s_1, \ldots, s_k\}$. The following (almost) now holds

$$X \models \Big(\bigwedge_{i \leq n} \top \subseteq q_i\Big) \vee \Big(\bigwedge_{i \leq n} \top \subseteq q_i\Big) \quad \text{iff} \quad \text{answer to set splitting is yes.}$$

Idea of the reduction from set splitting

Let $\mathcal{F} = \{Q_1, \ldots Q_n\}$, $S = \{s_i, \ldots, s_k\}$ be an instance of the problem.

- Consider each point s_i as an assignment s.t. $s_i(p_i) = 1$ and $s_i(p_j) = 0$.
- Sets Q_j are encoded s.t. $s_i(q_j) = 1$ iff $s_i \in Q_j$.

Define $X := \{s_1, \ldots, s_k\}$. The following (almost) now holds

$$X \models \Big(\bigwedge_{i \leq n} \top \subseteq q_i\Big) \vee \Big(\bigwedge_{i \leq n} \top \subseteq q_i\Big) \quad \text{iff} \quad \text{answer to set splitting is yes.}$$

$\mathrm{VAL}(\mathsf{ML}(\subseteq))$ is coNEXPTIME-hard

- DQBF is a NEXPTIME-complete generalisation of QBF.
- We give a reduction from the complement of DQBF to $\mathrm{VAL}(\mathsf{ML}(\subseteq))$.
- The proof: in the ArXiv-version of the paper.

$\mathrm{VAL}(\mathsf{ML}(\subseteq))$ is coNEXPTIME-hard

- DQBF is a NEXPTIME-complete generalisation of QBF.
- We give a reduction from the complement of DQBF to $\mathrm{VAL}(\mathsf{ML}(\subseteq))$.
- The proof: in the ArXiv-version of the paper.

What did we show?

	Satisfiability		Validity		Model checking	
	strict	lax	strict	lax	strict	lax
PL	— NP [Cook 71, Levin 73] —		—— coNP [Cook 71, Levin 73] ——		— NC^1 [Buss 87] —	
PL(⊆)	EXPTIME [†]	EXPTIME [†]	coNP	coNP [‡]	NP	P
ML	—— PSPACE [Ladner 77] ——		——— PSPACE [Ladner 77] ———		P [Clarke et al. 86]	
ML(⊆)	EXPTIME [†]	EXPTIME [†]	coNEXPTIME-h	coNEXPTIME-h	NP	P

† Hella et al. 2015, ‡ Hannula et al. 2015

What did we show?

THANKS!

	Satisfiability		Validity		Model checking	
	strict	lax	strict	lax	strict	lax
PL	— NP [Cook 71, Levin 73] —		—— coNP [Cook 71, Levin 73] ——		— NC^1 [Buss 87] —	
PL(⊆)	EXPTIME [†]	EXPTIME [†]	coNP	coNP [‡]	NP	P
ML	—— PSPACE [Ladner 77] ——		——— PSPACE [Ladner 77] ———		P [Clarke et al. 86]	
ML(⊆)	EXPTIME [†]	EXPTIME [†]	coNEXPTIME-h	coNEXPTIME-h	NP	P

† Hella et al. 2015, ‡ Hannula et al. 2015

References

- Pietro Galliani and Lauri Hella, Inclusion logic and fixed point logic, proceedings of *CSL 2013*.
- Lauri Hella, Antti Kuusisto, Arne Meier, and Heribert Vollmer, Modal inclusion logic: Being lax is simpler than being strict, proceedings of *MFCS 2015*.
- Miika Hannula, Juha Kontinen, Jonni Virtema, and Heribert Vollmer, Complexity of Propositional Independence and Inclusion Logic, proceedings of *MFCS 2015*.
- Miika Hannula, Martin Lück, Juha Kontinen, and Jonni Virtema, On quantified propositional logics and the exponential time hierarchy, proceedings of *GandALF 2016*.