Complexity and Expressivity of Propositional Logics with Team Semantics

ESSLLI 2024 course

Arne Meier[1] Jonni Virtema[2]

[1] Leibniz Universität Hannover, Germany
[2] University of Sheffield, UK

Version of 6th August 2024

Complexity and Expressivity of Propositional Logics with Team Semantics
Arne Meier, Jonni Virtema
7th of August

Lecture 3: Inclusion Logic

Literature: [Hel+19; Hel+20]

Union closure: $T_1 \models \varphi$, $T_2 \models \varphi \Rightarrow T_1 \cup T_2 \models \varphi$

Inspired by „inclusion dependencies“ from database theory.

$$T \models p_1 \cdots p_k \subseteq q_1 \cdots q_k \text{ iff } \forall\, u \in T\ \exists\, v \in T : u(\bar{p}) = v(\bar{q})$$

Lemma 32

$\mathrm{PL}[\subseteq]$ *is union closed.*

Proof by structural induction on φ.

IB $\varphi = p \in \mathrm{PROP}$. $T_1 \models p$ and $T_2 \models p \Rightarrow T_1 \cup T_2 \models p$.

$\varphi = \bar{p} \subseteq \bar{q}$. $T_1 \models \bar{p} \subseteq \bar{q}$ and $T_2 \models \bar{p} \subseteq \bar{q} \Rightarrow \forall u_1 \in T_1\ \exists v_1 \in T_1$ s.t. $u_1(\bar{p}) = v_1(\bar{q})$ and $\forall u_2 \in T_2\ \exists v_2 \in T_2$ s.t. $u_2(\bar{p}) = v_2(\bar{q})$

As a result: $\forall u \in T_1 \cup T_2\ \exists v \in T_1 \cup T_2$ s.t. $u(\bar{p}) = v(\bar{q})$. $\Rightarrow T_1 \cup T_2 \models \bar{p} \subseteq \bar{q}$

IS $\varphi = \alpha \wedge \beta$. By IH, we get that $T_1 \cup T_2 \models \alpha$ and $T_1 \cup T_2 \models \beta$. Hence, $T_1 \cup T_2 \models \alpha \wedge \beta$

$\varphi = \alpha \vee \beta$. By IH, we get that $T_1 \cup T_2 \models \alpha$ and $T_1 \cup T_2 \models \beta$. Hence, $T_1 \cup T_2 \models \alpha \vee \beta$ ∎

Validity in Team Semantics

Classical propositional validity is coNP-complete.

A formula φ is valid if $T \models \varphi$ for all teams T such that the propositions in φ are in the domain of T.

Problem: $\mathrm{VAL}(\mathcal{L})$ – the validity problem for logic $\mathcal{L}$
Input: a $\mathcal{L}$-formula φ
Question: Is φ valid?

Theorem 33

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

Hardness follows directly by classical validity which is coNP-complete.

Membership: 1) $PL[\subseteq]$ is union closed, so it suffices to look at singleton teams.

2) Say $\{s\}$ is some singleton team and $\bar{p} \subseteq \bar{q}$ is part of φ.

$\{s\} \models \bar{p} \subseteq \bar{q}$ iff $\bigwedge_{1 \le i \le k} p_i \leftrightarrow q_i$ is satisfied by s

Now, eliminate all inclusion atoms in φ. Call the new formula φ^*.

3) Now, $\varphi \in \mathrm{VAL}(PL[\subseteq])$ iff φ^* is a valid pop. formula.

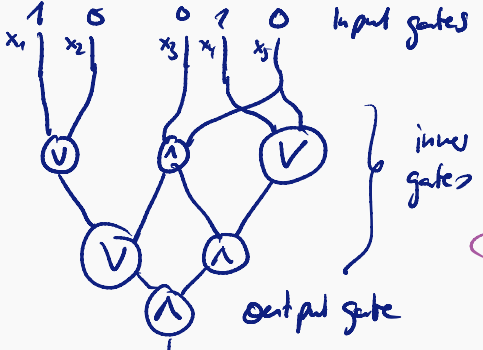
This reduction shows $\mathrm{VAL}(PL[\subseteq]) \in$ coNP. □

Foundations: Monotone circuit value problem

A 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.



formula

$$\big((x_1 \vee x_2) \vee (x_3 \wedge x_5)\big) \wedge \big((x_3 \wedge x_5) \wedge (x_4 \vee x_5)\big)$$

Foundations: Monotone circuit value problem

A 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.

Problem: MCVP — monotone circuit value problem
Input: a monotone circuit C and an input $b_1, \ldots, b_n \in \{0, 1\}$
Question: is the output of the circuit 1

Proposition 34 ([Gol77])

MCVP *is* P*-complete w.r.t.* $\leq_m^{\log}$*-reductions.*

Model-Checking for Inclusion Logic

Theorem 35 ([Hel+19, Thm. 3.5])

$\mathrm{PL}[\subseteq]\text{-}\mathrm{MC}$ *is* P*-complete.*

Model-Checking for Inclusion Logic

Theorem 35 ([Hel+19, Thm. 3.5])

$\mathrm{PL}[\subseteq]$-MC *is* P*-complete.*

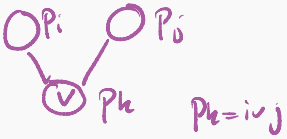
Ideas:

Lower bound: reduce from MCVP

Upper bound: use a labelling algorithm to compute a maximum satisfying team

P-hardness: Idea of the reduction from MCVP to $\mathrm{PL}[\subseteq]$-MC

- gate $g_i \rightsquigarrow$ assignment s_i
- proposition p_i for each gate g_i (where g_0 is the output gate), $p_\bot$ and $p_\top$
- special propositions $p_{k=i\vee j}$ for disjunction gates



P-hardness: Idea of the reduction from MCVP to $\mathrm{PL}[\subseteq]\text{-}\mathrm{MC}$

- gate $g_i \rightsquigarrow$ assignment s_i
- proposition p_i for each gate g_i (where g_0 is the output gate), $p_\bot$ and $p_\top$
- special propositions $p_{k=i\vee j}$ for disjunction gates
- $s_i \in T$ if g_i has value 1

"correct gate"

$$s_i(p) := \begin{cases} 1 & \text{if } p = p_i \text{ or } p = p_\top, \\ 1 & \text{if } p = p_{k=i\vee j} \text{ or } p = p_{k=j\vee i} \text{ for some } j, k \leq m, \\ 0 & \text{otherwise.} \end{cases}$$

"gate assignment"

P-hardness: Idea of the reduction from MCVP to $\mathrm{PL}[\subseteq]\text{-}\mathrm{MC}$

- gate $g_i \rightsquigarrow$ assignment s_i
- proposition p_i for each gate g_i (where g_0 is the output gate), $p_\bot$ and $p_\top$
- special propositions $p_{k=i\vee j}$ for disjunction gates
- $s_i \in T$ if g_i has value 1

$$s_i(p) := \begin{cases} 1 & \text{if } p = p_i \text{ or } p = p_\top, \\ 1 & \text{if } p = p_{k=i\vee j} \text{ or } p = p_{k=j\vee i} \text{ for some } j, k \leq m, \\ 0 & \text{otherwise.} \end{cases}$$

- $s_\bot(p) = 1$ iff $p = p_\bot$ or $p = p_\top$ (no other s_i maps $p_\bot$ to 1)
- create a formula φ_C that quantifies truth value of each gate and ensures correct propagation

More details: P-hardness of $\mathrm{PL}[\subseteq]$-MC.

After skipping some technicalities we arrive at

g_0 is output gate

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

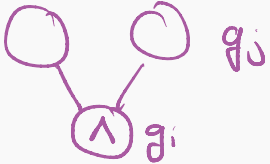
More details: P-hardness of $\mathrm{PL}[\subseteq]$-MC.

After skipping some technicalities we arrive at

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

Recall: gates that are in the team T have a value 1.

Express gate properties:



$$\begin{aligned} \psi_{\mathrm{out}=1} &:= p_\top \subseteq p_0, \\ \psi_\wedge &:= \bigwedge \{ p_i \subseteq p_j \mid (g_j, g_i) \in E \text{ and } \alpha(g_i) = \wedge \}, \\ \psi_\vee &:= \bigwedge \{ p_k \subseteq p_{k=i\vee j} \mid i < j, (g_i, g_k) \in E, (g_j, g_k) \in E, \text{ and } \alpha(g_k) = \vee \} \end{aligned}$$

Encoding MCVP: the final puzzle pieces

More truth about the team:

"input gates with input 1"

$$T := \{s_i \mid \alpha(g_i) \in \{\wedge, \vee\}\} \cup \{s_i \mid \alpha(g_i) \in \{x_i \mid b_i = 1\}\} \cup \{s_\perp\}$$

Encoding MCVP: the final puzzle pieces

More truth about the team:

$$T := \{s_i \mid \alpha(g_i) \in \{\wedge, \vee\}\} \cup \{s_i \mid \alpha(g_i) \in \{x_i \mid b_i = 1\}\} \cup \{s_\perp\}$$

"Wide circuit"

Now we claim that

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

Z Y

$T = Z \cup Y$.

Y : The correct circuit computation and output 1

Z : The "wrong" rest

Encoding MCVP: the final puzzle pieces

More truth about the team:

$$T := \{s_i \mid \alpha(g_i) \in \{\wedge, \vee\}\} \cup \{s_i \mid \alpha(g_i) \in \{x_i \mid b_i = 1\}\} \cup \{s_\bot\}$$

Now we claim that

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

Crux 1: split requires guessing a team Y for the right disjunct that encodes the valuation of the circuit.

Encoding MCVP: the final puzzle pieces

More truth about the team:

$$T := \{s_i \mid \alpha(g_i) \in \{\wedge, \vee\}\} \cup \{s_i \mid \alpha(g_i) \in \{x_i \mid b_i = 1\}\} \cup \{s_\perp\}$$

Now we claim that

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

Crux 1: split requires guessing a team Y for the right disjunct that encodes the valuation of the circuit.
Crux 2: $\neg p_\perp$ and $s_\perp$ ensure that Y is nonempty and deal with the propagation of the value 0 by the subformulae of the form $p_i \subseteq p_j$.

Computing a Maximum Satisfying Team

By maxsub(T, φ), we denote the maximum subteam T' of T such that $T' \models \varphi$, i.e., for all $T' \subsetneq T'' \subseteq T$, we have $T'' \not\models \varphi$.
Union closure of $\mathrm{PL}[\subseteq]$ ensures that this always exists.

Lemma 36 (for a proof, see [Hel+19, Lemma 5.1])

If φ is a proposition symbol, its negation, or an inclusion atom, then maxsub(T, φ) can be computed in polynomial time with respect to $|T| + |\varphi|$.

Computing a Maximum Satisfying Team

By maxsub(T, φ), we denote the maximum subteam T' of T such that $T' \models \varphi$, i.e., for all $T' \subsetneq T'' \subseteq T$, we have $T'' \not\models \varphi$.
Union closure of $\mathrm{PL}[\subseteq]$ ensures that this always exists.

Lemma 36 (for a proof, see [Hel+19, Lemma 5.1])

If φ is a proposition symbol, its negation, or an inclusion atom, then maxsub(T, φ) can be computed in polynomial time with respect to $|T| + |\varphi|$.

Interesting case: inclusion atoms (edges between assignments when agree on some variables; successively delete vertices with out-degree 0).

P-algorithm for $\mathrm{PL}[\subseteq]$-MC

Important properties:

- Each team T has a unique maximal subteam satisfying a given formula φ.
- For literals $\mathrm{maxsub}(T, \varphi)$ is computable in polynomial time (Lemma 36).

$\neg p, p, \bar{p} \subseteq \bar{q}$

P-algorithm for $\mathrm{PL}[\subseteq]$-MC

Important properties:

- Each team T has a unique maximal subteam satisfying a given formula φ.
- For literals $\mathrm{maxsub}(T, \varphi)$ is computable in polynomial time (Lemma 36).

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

1. Build the syntactic tree of φ and label each of its nodes with T.
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.

P-algorithm for $\mathrm{PL}[\subseteq]$-MC

Important properties:

- Each team T has a unique maximal subteam satisfying a given formula φ.
- For literals $\mathrm{maxsub}(T, \varphi)$ is computable in polynomial time (Lemma 36).

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

1. Build the syntactic tree of φ and label each of its nodes with T.
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.

Membership in P: more formally

approach: use a labelling function f_i of occurrences of subformulae of input φ and start with $f_0(\psi) = T$ for every sub-occurrence ψ

Membership in P: more formally

approach: use a labelling function f_i of occurrences of subformulae of input φ and start with $f_0(\psi) = T$ for every sub-occurrence ψ

bottum-up part (odd i):

- for literals ψ: $f_i(\psi) \coloneqq \mathsf{maxsub}(f_{i-1}(\psi), \psi)$
- $f_i(\psi \wedge \theta) \coloneqq f_i(\psi) \cap f_i(\theta)$
- $f_i(\psi \vee \theta) \coloneqq f_i(\psi) \cup f_i(\theta)$

Membership in P: more formally

approach: use a labelling function f_i of occurrences of subformulae of input φ and start with $f_0(\psi) = T$ for every sub-occurrence ψ

bottum-up part (odd i):

- for literals ψ: $f_i(\psi) \coloneqq \text{maxsub}(f_{i-1}(\psi), \psi)$
- $f_i(\psi \wedge \theta) \coloneqq f_i(\psi) \cap f_i(\theta)$
- $f_i(\psi \vee \theta) \coloneqq f_i(\psi) \cup f_i(\theta)$

top-down part (even $i > 0$):

- If $\psi = \theta \wedge \gamma$, let $f_i(\theta) \coloneqq f_i(\gamma) \coloneqq f_i(\theta \wedge \gamma)$.
- If $\psi = \theta \vee \gamma$, let $f_i(\theta) \coloneqq f_{i-1}(\theta) \cap f_i(\theta \vee \gamma)$ and $f_i(\gamma) \coloneqq f_{i-1}(\gamma) \cap f_i(\theta \vee \gamma)$.

Membership in P: more formally

approach: use a labelling function f_i of occurrences of subformulae of input φ and start with $f_0(\psi) = T$ for every sub-occurrence ψ

bottum-up part (odd i):

- for literals ψ: $f_i(\psi) := \text{maxsub}(f_{i-1}(\psi), \psi)$
- $f_i(\psi \wedge \theta) := f_i(\psi) \cap f_i(\theta)$
- $f_i(\psi \vee \theta) := f_i(\psi) \cup f_i(\theta)$

top-down part (even $i > 0$):

- If $\psi = \theta \wedge \gamma$, let $f_i(\theta) := f_i(\gamma) := f_i(\theta \wedge \gamma)$.
- If $\psi = \theta \vee \gamma$, let $f_i(\theta) := f_{i-1}(\theta) \cap f_i(\theta \vee \gamma)$ and $f_i(\gamma) := f_{i-1}(\gamma) \cap f_i(\theta \vee \gamma)$.

Claim: $f_\infty(\varphi) = T$ iff $T \models \varphi$

$f_{i+1} \subseteq f_i$

Satisfiability for Inclusion Logic

Theorem 37 ([Hel+20, Cor. 3.6])

$\mathrm{PL}[\subseteq]$-SAT *is* EXP*-complete.*

Short ideas:

Propositional Dynamic Logic

Upper bound: equivalence preserving translation to SAT in PDL with global and converse modalities

Lower bound: reduce from succinct Path-$\mathrm{Systems}$ variant

Satisfiability for Inclusion Logic

Theorem 37 ([Hel+20, Cor. 3.6])

$\mathrm{PL}[\subseteq]$-SAT *is* EXP-*complete.*

Short ideas:

Upper bound: equivalence preserving translation to SAT in PDL with global and converse modalities

Lower bound: reduce from succinct $\mathrm{Path\text{-}Systems}$ variant (our focus)

L P-complete

Persistent subsets

Definition 38

Let $\mathfrak{A} = (A, S)$ be a structure with $A = \{1, \ldots, n\}$ and $S \subseteq A^3$. A subset P of A is S-persistent if it satisfies the condition

$(*)$ if $i \in P$, then there are $j, k \in P$ such that $(i, j, k) \in S$.

Problem: PER
Input: structures $\mathfrak{A} = (A, S)$ with $A = \{1, \ldots, n\}$ and $S \subseteq A^3$
Question: exists some S-persistent set $P \subseteq A$ such that $n \in P$

Theorem 39 (closely related to $\mathrm{PathSystems}$ [GHR95, p. 171])

PER *is* P*-complete.*

A succinct variant of PER

- represent structures $\mathfrak{A} = (A, S)$ by Boolean circuits C with inputs of length 3ℓ
- $\mathfrak{A} = (A, S) \overset{C}{\rightsquigarrow} (A_C, S_C)$ with $A_C = \{1, \ldots, 2^\ell\}$
- for all $i, j, k \in A$, let $(i, j, k) \in S_C$ if and only if C accepts the input tuple

$$(a_1, \ldots, a_\ell, b_1, \ldots, b_\ell, c_1, \ldots, c_\ell) \in \{0,1\}^{3\ell},$$

where $i = \mathrm{bin}(a_1 \ldots a_\ell)$, $j = \mathrm{bin}(b_1 \ldots b_\ell)$ and $k = \mathrm{bin}(c_1 \ldots c_\ell)$.

Problem: S-PER
Input: Boolean circuits C with inputs of length 3ℓ , $\mathfrak{A} = (A, S)$
Question: exists some S_C-persistent set $P \subseteq A_C$ such that $2^\ell \in P$

Theorem 40 ([Hel+19, Lem. 3.4])

S-PER *is* EXP*-hard with respect to* $\leq_m^p$*-reductions.*

EXP-**hardness of** $\mathrm{PL}[\subseteq]$**: prerequisites**

To show: $\text{S-PER} \leq_m^p \mathrm{PL}[\subseteq]\text{-SAT}$.

Notation: $T(p_1, \ldots, p_n) \coloneqq \{(s(p_1), \ldots, s(p_n)) \in \{0,1\}^n \mid s \in T\}$

EXP-**hardness of** $\mathrm{PL}[\subseteq]$**: prerequisites**

To show: $\text{S-PER} \leq_m^p \mathrm{PL}[\subseteq]\text{-SAT}$.
Notation: $T(p_1, \ldots, p_n) \coloneqq \{(s(p_1), \ldots, s(p_n)) \in \{0,1\}^n \mid s \in T\}$

Note that the semantics of inclusion atoms can now be expressed as

$$T \models p_1 \cdots p_n \subseteq q_1 \cdots q_n \iff T(p_1, \ldots, p_n) \subseteq T(q_1, \ldots, q_n).$$

Encoding S-PER into PL[⊆]

C is a Boolean circuit with 3ℓ input gates ordered $g_1, \ldots, g_m$ such that $g_1, \ldots, g_{3\ell}$ are the input gates and g_m is the output gate.

- fix propositions p_i for each gate g_i
- define for each gate a $\mathrm{PL}[\subseteq]$ formula θ_i:

$$\theta_i = \begin{cases} p_i \leftrightarrow \neg p_j & \text{if } g_i \text{ is a NOT gate with input } g_j \\ p_i \leftrightarrow (p_j \wedge p_k) & \text{if } g_i \text{ is an AND gate with inputs } g_j \text{ and } g_k \\ p_i \leftrightarrow (p_j \vee p_k) & \text{if } g_i \text{ is an OR gate with inputs } g_j \text{ and } g_k \end{cases}$$

Note: $\leftrightarrow$ is usual shorthand for flat formulas

↳ "$\vee$" coincides with classical "or"-semantics

Encoding S-PER into PL[⊆]

C is a Boolean circuit with 3ℓ input gates ordered $g_1, \ldots, g_m$ such that $g_1, \ldots, g_{3\ell}$ are the input gates and g_m is the output gate.

- fix propositions p_i for each gate g_i
- define for each gate a $\mathrm{PL}[\subseteq]$ formula θ_i:

$$\theta_i = \begin{cases} p_i \leftrightarrow \neg p_j & \text{if } g_i \text{ is a NOT gate with input } g_j \\ p_i \leftrightarrow (p_j \wedge p_k) & \text{if } g_i \text{ is an AND gate with inputs } g_j \text{ and } g_k \\ p_i \leftrightarrow (p_j \vee p_k) & \text{if } g_i \text{ is an OR gate with inputs } g_j \text{ and } g_k \end{cases}$$

Note: $\leftrightarrow$ is usual shorthand for flat formulas

Then: $\psi_C := \left(\bigwedge_{3\ell+1 \le i \le m} \theta_i\right) \wedge p_m$. (truth values of p_i match acc. computation of C)

"inner gates" "output is 1"

Encoding persistency into the formula

Input gate propositions: $p_1, \ldots, p_\ell, \underbrace{p_{\ell+1}, \ldots, p_{2\ell}}_{=:q_1,\ldots,q_\ell}, \underbrace{p_{2\ell+1}, \ldots, p_{3\ell}}_{=:r_1,\ldots,r_\ell}$

Encoding persistency into the formula

Input gate propositions: $p_1, \ldots, p_\ell, \underbrace{p_{\ell+1}, \ldots, p_{2\ell}}_{=:q_1,\ldots,q_\ell}, \underbrace{p_{2\ell+1}, \ldots, p_{3\ell}}_{=:r_1,\ldots,r_\ell}$ The final formula:

"Correct acc. comp"

$$\varphi_C := \psi_C \wedge q_1 \cdots q_\ell \subseteq p_1 \cdots p_\ell \wedge r_1 \cdots r_\ell \subseteq p_1 \cdots p_\ell \wedge p_m \cdots p_m \subseteq p_1 \cdots p_\ell$$

Claim: C is a positive instance of $\mathrm{S\text{-}PER}$ if and only if φ_C is satisfiable.

Encoding persistency into the formula

Input gate propositions: $p_1, \ldots, p_\ell, \underbrace{p_{\ell+1}, \ldots, p_{2\ell}}_{=:q_1,\ldots,q_\ell}, \underbrace{p_{2\ell+1}, \ldots, p_{3\ell}}_{=:r_1,\ldots,r_\ell}$ The final formula:

$$\varphi_C \coloneqq \psi_C \wedge q_1 \cdots q_\ell \subseteq p_1 \cdots p_\ell \wedge r_1 \cdots r_\ell \subseteq p_1 \cdots p_\ell \wedge p_m \cdots p_m \subseteq p_1 \cdots p_\ell$$

Claim: C is a positive instance of $\mathrm{S\text{-}PER}$ if and only if φ_C is satisfiable.

We will prove only $\Rightarrow$. For the other direction, we define the S_C-persistent set as

$$\{\mathrm{bin}(a_1 \ldots a_\ell) \mid (a_1, \ldots, a_\ell) \in T(p_1, \ldots, p_\ell)\}.$$

"⇒" of the claim

C is a pos. instance for S-PER.
$\Rightarrow$ there is a S_C-persistent set $P \subseteq A_C$ s.t. $2^\ell \in P$ (*)
Let T be a tree s.t. $(a_1, \dots, a_m) \in T$ iff $bin(a_1 \dots a_m), bin(a_{\ell+1} \dots a_{2\ell}), bin(a_{2\ell+1} \dots a_{3\ell}) \in P$.

To show: $T \models \varphi_C$.

1) $T \models \varphi_C$ ✓ as we defined T to suffice persistency.
2) $T \models q_1 \dots q_\ell \subseteq p_1 \dots p_\ell$. For that assume $(b_1, \dots, b_\ell) \in T(q_1, \dots, q_\ell)$
   Then: $i := bin(b_1 \dots b_\ell) \in P$ and $j, k \in P$ s.t. $(i,j,k) \in S_C$ true due to (*)
   Thus: we have $(a_1, \dots, a_m)$ corresp. to an accepting computation s.t.
   $(a_1, \dots, a_\ell) = (b_1, \dots, b_\ell)$, $j = bin(b_{\ell+1} \dots b_{2\ell})$, $k = bin(a_{2\ell+1} \dots a_{3\ell})$
   $\Rightarrow (a_1, \dots, a_m) \in T$ and $(b_1, \dots, b_\ell) \in T(p_1, \dots, p_\ell)$

3) Similarly for $T \models r_1 \cdots r_\ell \subseteq p_1 \cdots p_\ell$

4) Since $T \models p_m \Rightarrow T(p_m, \dots, p_m) = \{(1, \dots, 1)\}$

Since $2^\ell = bin(1, \dots, 1) \in P$ there is $(a_1, \dots, a_m) \in T$ s.t.

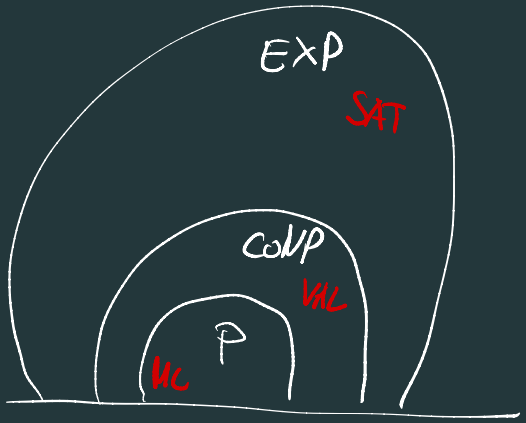
$(a_1, \dots, a_\ell) = (1, \dots, 1)$.

Thus $(1, \dots, 1) \in T(p_1, \dots, p_\ell)$ and $T \models p_m \cdots p_m \subseteq p_1 \cdots p_\ell$.

□

Conclusion of Lecture 3

- $\mathrm{PL}[\subseteq]$-MC is P-complete.
- $\mathrm{PL}[\subseteq]$-SAT is EXP-complete.
- $\mathrm{PL}[\subseteq]$-VAL is coNP-complete.



Complexity and Expressivity of Propositional Logics with Team Semantics
Arne Meier, Jonni Virtema
8th of August

Lecture 4: Complexity of propositional dependence logic and beyond

Literature: [Vir17; Han+18]

Bibliography i

[BRV01] Patrick Blackburn, Maarten de Rijke and Yde Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001. DOI: 10.1017/CBO9781107050884.

[CES86] E. Clarke, E. Allen Emerson and A. Sistla. 'Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications'. In: *ACM Transactions on Programming Languages and Systems* 8.2 (1986), pp. 244–263.

[Coo71] Stephen A. Cook. 'The Complexity of Theorem-Proving Procedures'. In: *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*. Ed. by Michael A. Harrison, Ranan B. Banerji and Jeffrey D. Ullman. ACM, 1971, pp. 151–158. DOI: 10.1145/800157.805047. URL: https://doi.org/10.1145/800157.805047.

Bibliography ii

[EFT94] Heinz-Dieter Ebbinghaus, Jörg Flum and Wolfgang Thomas. *Mathematical logic (2. ed.)* Undergraduate texts in mathematics. Springer, 1994.

[EJ99] E. Allen Emerson and Charanjit S. Jutla. 'The Complexity of Tree Automata and Logics of Programs'. In: *SIAM J. Comput.* 29.1 (1999), pp. 132–158.

[ES84] E. Allen Emerson and A. Prasad Sistla. 'Deciding Full Branching Time Logic'. In: *Inf. Control.* 61.3 (1984), pp. 175–201.

[FL79] Michael J. Fischer and Richard E. Ladner. 'Propositional Dynamic Logic of Regular Programs'. In: *J. Comput. Syst. Sci.* 18.2 (1979), pp. 194–211.

[GHR95] Raymond Greenlaw, H. James Hoover and Walter L. Ruzzo. *Limits to Parallel Computation: P-completeness Theory*. New York, NY, USA: Oxford University Press, Inc., 1995. ISBN: 0-19-508591-4.

Bibliography iii

[Gol77] L. M. Goldschlager. 'The monotone and planar circuit value problems are log-space complete for P'. In: *SIGACT News* 9 (1977), pp. 25–29.

[Gut+22] Jens Oliver Gutsfeld, Arne Meier, Christoph Ohrem and Jonni Virtema. 'Temporal Team Semantics Revisited'. In: *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*. Ed. by Christel Baier and Dana Fisman. ACM, 2022, 44:1–44:13. DOI: 10.1145/3531130.3533360. URL: https://doi.org/10.1145/3531130.3533360.

[Han+18] Miika Hannula, Juha Kontinen, Jonni Virtema and Heribert Vollmer. 'Complexity of Propositional Logics in Team Semantic'. In: *ACM Trans. Comput. Log.* 19.1 (2018), 2:1–2:14.

Bibliography iv

[Han19] Miika Hannula. 'Validity and Entailment in Modal and Propositional Dependence Logics'. In: *Logical Methods in Computer Science* Volume 15, Issue 2 (Apr. 2019). DOI: 10.23638/LMCS-15(2:4)2019. URL: https://lmcs.episciences.org/5403.

[Hel+14] Lauri Hella, Kerkko Luosto, Katsuhiko Sano and Jonni Virtema. 'The Expressive Power of Modal Dependence Logic'. In: *Advances in Modal Logic*. College Publications, 2014, pp. 294–312.

[Hel+19] Lauri Hella, Antti Kuusisto, Arne Meier and Jonni Virtema. 'Model checking and validity in propositional and modal inclusion logics'. In: *J. Log. Comput.* 29.5 (2019), pp. 605–630.

[Hel+20] Lauri Hella, Antti Kuusisto, Arne Meier and Heribert Vollmer. 'Satisfiability of Modal Inclusion Logic: Lax and Strict Semantics'. In: *ACM Trans. Comput. Log.* 21.1 (2020), 7:1–7:18.

Bibliography v

[HS15] Lauri Hella and Johanna Stumpf. 'The expressive power of modal logic with inclusion atoms'. In: *GandALF*. Vol. 193. EPTCS. 2015, pp. 129–143.

[KV85] Gabriel M. Kuper and Moshe Y. Vardi. 'On the Expressive Power of the Logical Data Model (Preliminary Report)'. In: *SIGMOD Conference*. ACM Press, 1985, pp. 180–187.

[Lev73] Leonid A. Levin. 'Universal sequential search problems'. In: *Problemy Peredachi Informatsii* 9.3 (1973).

[Loh12] Peter Lohmann. 'Computational Aspects of Dependence Logic'. PhD thesis. Leibniz Universität Hannover, 2012. arXiv: 1206.4564. URL: http://arxiv.org/abs/1206.4564.

[LV19] Martin Lück and Miikka Vilander. 'On the Succinctness of Atoms of Dependency'. In: *Log. Methods Comput. Sci.* 15.3 (2019).

Bibliography vi

[Pap07] Christos H. Papadimitriou. *Computational complexity*. Academic Internet Publ., 2007.

[Pra80] V. R. Pratt. 'A near-optimal method for reasoning about action'. In: *Journal of Computer and System Sciences* 20.2 (1980), pp. 231–254.

[SC85] A. Prasad Sistla and Edmund M. Clarke. 'The Complexity of Propositional Linear Temporal Logics'. In: *J. ACM* 32.3 (1985), pp. 733–749.

[Sch02] P. Schnoebelen. 'The Complexity of Temporal Logic Model Checking'. In: *Advances in Modal Logic*. Vol. 4. 2002, pp. 393–436.

[Sip97] Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.

[Var09] Moshe Y. Vardi. 'From Philosophical to Industrial Logics'. In: *ICLA*. Vol. 5378. Lecture Notes in Computer Science. Springer, 2009, pp. 89–115.

Bibliography vii

[Vir+21] Jonni Virtema, Jana Hofmann, Bernd Finkbeiner, Juha Kontinen and Fan Yang. 'Linear-Time Temporal Logic with Team Semantics: Expressivity and Complexity'. In: *FSTTCS*. Vol. 213. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 52:1–52:17.

[Vir17] Jonni Virtema. 'Complexity of validity for propositional dependence logics'. In: *Inf. Comput.* 253 (2017), pp. 224–236.

[YV17] Fan Yang and Jouko Väänänen. 'Propositional team logics'. In: *Ann. Pure Appl. Log.* 168.7 (2017), pp. 1406–1441. DOI: 10.1016/J.APAL.2017.01.007. URL: https://doi.org/10.1016/j.apal.2017.01.007.