Complexity and Expressivity of Propositional Logics with Team Semantics

Lecture Notes ESSLLI 2024 course

Arne Meier[1] Jonni Virtema[2]

Version of 2024-08-08

[1] Leibniz Universität Hannover
[2] University of Sheffield

Preface

Hello everyone,

this course is about team semantics. This formalism is a generalisation of standard Tarskian semantics. We will focus in this course on propositional logic, although there is also a first-order variant. The course will consist of five lectures, each focusing on different aspects of subjective interest. It will give a small insight into the research area around team semantics.

1. Syntax and semantics, properties, problems.
2. Expressiveness.
3. Inclusion Logic: P-complete MC, coNP-complete VAL.
4. Dependence. MC(PDL) is NP-complete. DQBF, VAL(PDL) is NEXP-complete.
5. Hyperproperties, temporal aspects. TeamLTL(inclusion, dep) is undecidable.

For each lecture, we give some references that we think are interesting and might be helpful for further reading. Feel free to approach us in the lecture or send us an email if you have any questions. We are happy to help.

Hannover & Sheffield, in July 2024 Arne Meier & Jonni Virtema

Contents

Preliminary skirmish

Organisational information about the course

Material for the course:

- Lecture notes
- Slides (in advance) and writings into slides (afterwards)
- Webpage: `https://www.thi.uni-hannover.de/de/esslli24`

About the lecturers

Arne Meier (Leibniz University Hannover)
Research Interests: Complexity Theory, Foundations of AI, Non-Classical Logics, Enumeration
`https://arnemeier.github.io`

Jonni Virtema (University of Sheffield)
Research Interests: Finite Model Theory, Temporal Logics for Hyperproperties, Logical Foundations of Neural Networks, Complexity Theory.
`http://www.virtema.fi/`

Prerequisites and requirements

- Complexity theory foundations, e.g., [Pap07; Sip97]
- Propositional Logic foundations, e.g., [EFT94]
- Modal Logic (only relevant for last lecture), e.g., [BRV01]

Course outline

Monday, 5th of August Syntax and Semantics, Properties, Problems.

Tuesday, 6th of August Expressivity and succintness

Wednesday, 7th of August Inclusion Logic: P-complete MC, coNP-complete VAL

Thursday, 8th of August Dependence. Show MC(PDL) is NP-complete. DQBF, VAL(PDL) is NEXP-complete

Friday, 9th of August Hyperproperties, Temporal Aspects. TeamLTL(inclusion, dep) is undecidable.

1 Propositional Logics with Team Semantics

Literature: [YV17]

In 2006, Dependence Logic was introduced by Jouko Väänänen in 2007 [Vää07]. It is a logic that extends first-order logic with a new operator (the dependence atom) for expressing dependence between variables. At its heart lies the concept of team semantics, which is a generalization of the standard Tarskian semantics of first-order logic to sets of assignments.

In this ESSLLI course, we will study propositional logics with team semantics. At first, we will give some intuition about these concepts and then introduce the syntax and semantics of the logics we will study.

Dependence and independence

What means "x depends on y" or "x and y are independent"?

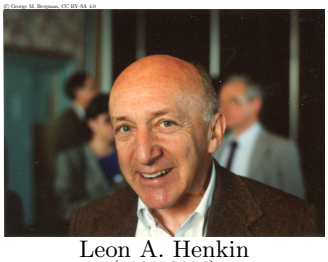
Compare it to: "x divides y"
Here: fix structure $\mathcal{A}$, with well-defined division and find an assignment $s\colon \{x, a\} \to A$.
Then: Check Tarskian semantics of $\mathcal{A} \models_s$"x divides y"
Caution: (In-)dependence is different. It does not manifest itself in single assignments, but in

- tables or relations
- sets of rounds of a game
- sets of assignments $\rightsquigarrow$ teams

(In-)Dependence Logics: Henkin-Quantifiers (1959)

$$\varphi = \begin{pmatrix} \forall x & \exists y \\ \forall u & \exists v \end{pmatrix} P(x, y, u, v)$$

Semantics: over Skolem functions or via games with imperfect information



Leon A. Henkin
(1921–2006)

$(A, P) \models \varphi$, if there are functions $f, g\colon A \to A$ such that for all $a, c \in A$

$$P(a, f(a), c, g(c))$$

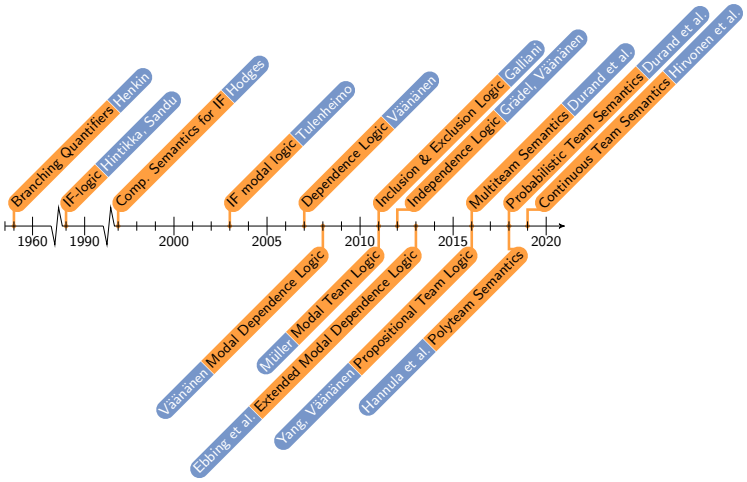
(In-)Dependence Logics: Independence-friendly logic (1989)

- First logic with quantifiers that are annotated with independence
- Quantification: φ formula, x variable, W finite set of variables yields expressions $(\exists x/W)\varphi$ and $(\forall x/W)\varphi$
- Game-theoretic Semantics: In the evaluation game for $(\partial x/W)\varphi$, the value x has to be chosen independent of the values in W
- At two positions $((\partial x/W)\varphi, s)$ and $((\partial x/W)\varphi, s')$ with $s(y) \neq s'(y)$ for all $y \in W$, the same value for x has to be chosen

Jaakko Hintikka
(1929-2005)

Gabriel Sandu
(* 1954)

Dependence Logic: Historically



Jouko Väänänen
(* 1950)

Dependence Logic: A Bit of Motivation

Primary key

docent	time	room	lecture
Antti	09:00	A.10	Genetics
Antti	11:00	A.10	Biochemistry
Antti	15:00	B.20	Ecology
Jonni	10:00	C.30	Bio-LAB
Juha	10:00	C.30	Bio-LAB
Juha	13:00	A.10	Biochemistry
⋮	⋮	⋮	⋮

Task: Consistency check of a timetable.
$\{\texttt{docent}, \texttt{time}\}$ functionally determines $\{\texttt{room}, \texttt{lecture}\}$, where $\{\texttt{room}, \texttt{time}\}$ does not functionally determine $\{\texttt{docent}\}$.

Dependence Logic: Applications

Dependence Atom

- Models functional dependencies in *sets* of assignments
- Semantics: y depends on x, i.e., y is uniquely determined by x

x	y	z
0	1	0
0	1	1

$\models \mathrm{dep}(\{x\}; \{y\})$

x	y	z
0	1	0
0	0	1

$\not\models \mathrm{dep}(\{x\}; \{y\})$

Applications: Modelling of...

- database schemes
- deterministic behaviour
- specifications
- ...

Team-Based Propositional Logic

Definition 1. Let PROP be a countably infinite set of propositions.

- An assignment s is a mapping $s\colon \mathsf{PROP} \to \{0, 1\}$.

- Propositional Team Logic (PL):

$$\varphi ::= x \mid \neg x \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$$

  where $x \in \mathsf{PROP}$.

- A *team* over PROP is a set of assignments, i.e., an element of $\mathcal{P}(2^{\mathsf{PROP}})$

Propositional Team Semantics

Definition 2. Let T be a team and $\varphi, \psi \in \mathrm{PL}[\mathrm{dep}]$. We define $T \models \varphi$ recursively via:

$$\begin{array}{lll} T \models x & \text{iff} & s(x) = 1 \quad \forall s \in T, \\ T \models \neg x & \text{iff} & s(x) = 0 \quad \forall s \in T, \\ T \models \varphi \wedge \psi & \text{iff} & T \models \varphi \text{ and } T \models \psi, \\ T \models \varphi \vee \psi & \text{iff} & \exists T_1 \exists T_2 (T = T_1 \cup T_2) \text{ s.t. } T_1 \models \varphi \text{ and } T_2 \models \psi. \end{array}$$

Caution: Only atomic negation here.

Dependence Atoms

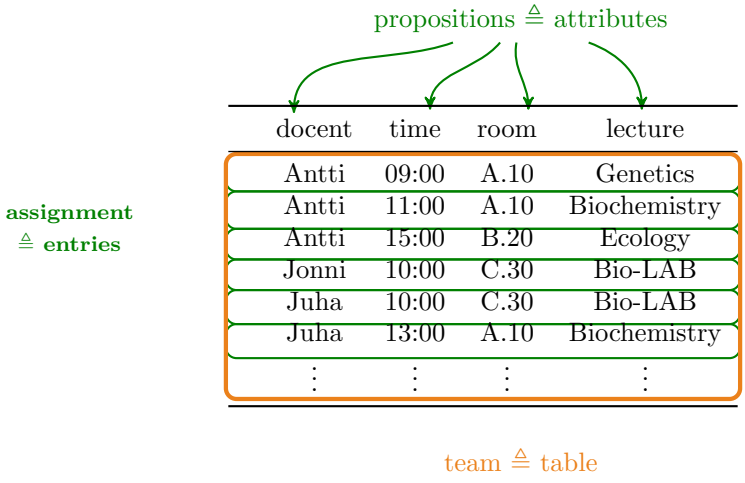
For assignment $s \colon \mathsf{PROP} \to \{0,1\}$ and $P \subseteq \mathsf{PROP}$, $s{\restriction}P$ is the assignment s restricted to P only. Let $P, Q \subseteq \mathsf{PROP}$.

$$T \models \mathrm{dep}(P;Q) \quad \text{iff} \quad \forall s,t \in T : s{\restriction}P = t{\restriction}P \Rightarrow s{\restriction}Q = t{\restriction}Q.$$

Notation: PL[dep] for propositional logic with dependence atoms.
Observations:

$$\begin{array}{lll} T \models \mathrm{dep}(;Q) & \text{iff} & \forall s,t \in T : s{\restriction}Q = t{\restriction}Q \quad \text{(Constancy Atom)}, \\ T \models \neg\mathrm{dep}(P,Q) & \text{iff} & T = \emptyset. \end{array}$$

Team Semantics from the Database Perspective



Back to the Initial Example

primary key

docent	time	room	lecture
Antti	09:00	A.10	Genetics
Antti	11:00	A.10	Biochemistry
Antti	15:00	B.20	Ecology
Jonni	10:00	C.30	Bio-LAB
Juha	10:00	C.30	Bio-LAB
Juha	13:00	A.10	Biochemistry
⋮	⋮	⋮	⋮

$\Rightarrow$ $\{\texttt{docent}, \texttt{time}\}$ functionally determines $\{\texttt{room}, \texttt{lecture}\}$.

How do you express this in PL[dep]?

$\mathrm{dep}(\{\texttt{docent}, \texttt{time}\}, \{\texttt{room}, \texttt{lecture}\})$

We only consider Propositional Team Logic here

Caution: encode all entries in binary (Propositional Logic vs. FO)

docent	room	time	lecture
Antti	A.10	09.00	Genetics
Antti	A.10	11.00	Biochemistry
Antti	B.20	15.00	Ecology
Jonni	C.30	10.00	Bio-Lab
Juha	C.30	10.00	Bio-Lab
Juha	A.10	13.00	Biochemistry

$i_1 i_2$	$r_1 r_2$	$t_1 t_2 t_3$	$c_1 c_2$
00	11	110	11
00	11	111	00
00	00	000	01
01	01	001	10
10	01	001	10
10	11	010	00

(Left) Sample database with 4 attributes and universe size 15.
(Right) Encoding with $\lceil \log_2(3) \rceil + \lceil \log_2(3) \rceil + \lceil \log_2(5) \rceil + \lceil \log_2(4) \rceil$-many propositions.

Interesting and Important Properties of such Logics

property	definition	dep	$\subseteq$	$\bot$	$\mid$
Downward closure	$T \models \varphi$ and $T' \subseteq T$ implies $T' \models \varphi$	✓	×	×	✓
Union closure	$T \models \varphi$ and $T' \models \varphi$ implies $T \cup T' \models \varphi$	×	✓	✓	×

All logics considered here are:

$$\text{flat: } T \models \varphi \iff \forall s \in T : \{s\} \models \varphi$$

and satisfy the

$$\text{Empty team property: } \emptyset \models \varphi.$$

Downward Closure of PL[dep]

Lemma 3. PL[dep] *is downward closed.*

Proof. We give a structural inductive proof for PL[dep] formulae of length.

Induction beginning.

- Let $\varphi = x$ be a proposition. If $T \models x$ and $T' \subseteq T$, then $T' \models x$ by semantics.
- Let $\varphi = \mathrm{dep}(P;Q)$. Then, by semantics, for all $s,t \in T$ with $s{\restriction}P = t{\restriction}P$ we have $s{\restriction}Q = t{\restriction}Q$. Clearly, this is still true for all subsets T' of T.

Induction step. Furthermore, let $T \models \varphi$. Do a case distinction according to φ. If φ is

- $\neg\psi$, then, as we only have atomic negation, $\psi \in \mathsf{PROP}$. As a result, $s(p) = 0$ for all $s \in T$. Accordingly, for all $T' \subseteq T$ we have that $T' \models \varphi$.
- $\psi \wedge \chi$, then $T \models \psi$ and $T \models \psi$. By IH, we get that for all $T' \subseteq T$ we have that $T' \models \varphi$.
- $\psi \vee \chi$, then there exists two teams T_1 and T_2 with $T = T_1 \cup T_2$ such that $T_1 \models \psi$ and $T_2 \models \chi$. By IH, all $T_1' \subseteq T_1$ and $T_2' \subseteq T_2$ satisfy ψ and χ, respectively. Thus, $T' \models \varphi$ where $T' = T_1' \cup T_2'$. □

There exists a different argumentation for the case of $\varphi \vee \psi$ that was pointed out by one of the participants (thank you for that) which is also correct and works as follows.

- $\psi \vee \chi$. Let $T \models \psi \vee \chi$. By semantics, we have that there exists split $T_1 \cup T_2 = T$ such that $T_1 \models \varphi$ and $T_2 \models \psi$. Consider any team $T' \subseteq T$. Then, IH gives us that $T' \cap T_1 \models \psi$ and $T' \cap T_2 \models \chi$. Hence, $(T' \cap T_1) \cup (T' \cap T_2) = T'$ by set theory. Thus, $T' \models \varphi \vee \psi$ for all teams $T' \subset T$ as T' was chosen arbitrarily.

In the following, we denote by $\mathbf{Vars}(\varphi)$ the set of all propositions occurring in φ.

Decision Problems

Problem: PL[dep]-MC — the model checking problem
Input: A PL[dep]-formula φ, a team T over $\mathbf{Vars}(\varphi)$
Question: Is $T \models \varphi$ true?

Problem: PL[dep]-SAT — the satisfiability problem
Input: A PL[dep]-formula φ
Question: Exists a non-empty team T over $\mathbf{Vars}(\varphi)$ with $T \models \varphi$?

Theorem 4 ([Loh12, Theorem 4.13], *Proof: Thursday*)**.** PL[dep]-MC *is* NP*-complete.*

We present the simplest complexity result of these two problems: satisfiability. Due to downward closure this problem can be reduced to satisfiability of PL.

Satisfiability Does not Become Harder Than in the Classical Case

Theorem 5 ([Coo71; Lev73])**.** PL[dep]-SAT *is* NP*-complete.*

Lemma 6 (follows from downward closure)**.** *A* PL[dep]*-formula is satisfiable if and only if a singleton team satisfies it.*

Note that any singleton team satisfies any dependence atom!

For $\varphi \in \mathrm{PL}[\mathrm{dep}]$, let φ^* denote the formula obtained from φ by substituting all dependence atoms with $\top$.

Lemma 7. *For any* $\varphi \in \mathrm{PL}[\mathrm{dep}]$ *and singleton team* $\{s\}$*, we have* $\{s\} \models \varphi \iff \{s\} \models \varphi^*$.

A straightforward induction proves the lemma.

Proof of Thm. 5. NP-hardness follows from NP-completeness of SAT(PL). Inclusion to NP follows from it as well via the above two lemmas: $\varphi \in \mathrm{PL}[\mathrm{dep}]$ is satisfiable iff $\varphi^* \in \mathrm{PL}$ is. □

Inclusion (Wednesday)

Inspired by "inclusion dependencies" from database theory.

Let $p_1, \ldots, p_k$ and $q_1, \ldots, q_k$ be propositions. Write $\bar{p}$, $\bar{q}$ for $p_1, \ldots, p_k$ and $q_1, \ldots, q_k$, respectively.

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

Theorem 8 ([Hel+20, Cor. 3.6])**.** PL[⊆]-SAT *is* EXP*-complete.*

Theorem 9 ([Hel+19, Thm. 13])**.** PL[⊆]-MC *is* P*-complete.*

Looking Beyond the Horizon: Independence

Caution: Also exists in stochastics; two events are *independent* if the occurrence of one does not influence the probability of the other occurring

But:

- logical independence compatible with this
- every possible pattern for (x, y) occurs, but how often does not matter
- knowing only x/y gives no information about the other

$$T \models p_1 \cdots p_k \perp_{r_1 \cdots r_\ell} q_1 \cdots q_n \text{ iff } \forall\, (u, v) \in T \times T \text{ s.t. } u(\bar{r}) = v(\bar{r}) \\ \exists\, w \in T : u(\bar{p}\bar{r}) = w(\bar{p}\bar{r}) \wedge w(\bar{q}) = v(\bar{q})$$

"the variables in $\bar{p}$ are completely independent of $\bar{q}$ for each constant value of $\bar{r}$"

Theorem 10 ([Han+18])**.** PL[⊥]-SAT *and* PL[⊥]-MC *are* NP*-complete.*

Note that it is possible to translate independence atoms to dependence atoms [Gal12].

Looking Beyond the Horizon: Exclusion

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

Theorem 11 (by vanilla SAT, [Coo71; Lev73])**.** PL[|]*-SAT is* NP*-complete.*

In the following, we will use a complexity class which makes use of the concept of oracle Turing machines. Intuitively, an oracle Turing machine is a classical Turing machine that has an additional oracle tape and three additional states. One to trigger the oracle question $q_?$, one for the positive oracle answer q_+ and one for the negative oracle answer q_-. An oracle call is usually charged one time unit. For further information on that topic consider the textbook of Sipser [Sip97].

Implication

A formula φ *entails* a formula ψ if and only if every team that satisfies φ also satisfies ψ, written $\varphi \models \psi$. A set of formulae Σ entails a formula φ if and only if every team that satisfies all formulae in Σ also satisfies φ, written $\Sigma \models \varphi$.

Problem: PL[dep]-IMP — the entailment problem for PL[dep]
Input: a set of PL[dep]-formulae Σ, a PL[dep]-formula φ
Question: Is $\Sigma \models \varphi$ true?

Theorem 12 ([Han19, Thm. 5.6, Thm. 6.1])**.** PL[dep]-IMP *is* $\mathsf{coNEXPTIME}^{\mathsf{NP}}$*-complete.*

As an idea for the membership result, we need to guess a team of potentially exponential size such that the left side is satisfied and the right side is falsified. For these, we need to model check also the formulae which can be done in NP (Thm. 4). Note that the result of the implication problem can also be strengthened to the extended modal dependence logic EMDL [Ebb+13].

Conclusion of Lecture 1

- Team semantics
- Dependence Atoms
- Inclusion, Exclusion, Independence
- Complexity of Satisfiability of PL[dep]
- Properties of PL[dep]

2 Expressive power of team-based logics

Literature: [YV17; Hel+14]

Next we will discuss about the experessive power of team based logics. What kind of properties can be expressed in these logics? Can we precisely characterise those properties that can be expressed in these logics?

How to characterise expressivity – Tarski's semantics

Definition 13. If φ is formula of propositional logic, with variables $p_1 \dots, p_n$, one can say that φ defines the n-ary Boolean function $f_\varphi : \{0,1\}^n \to \{0,1\}$ defined

$$s \mapsto s(\varphi),$$

where s is an assignment for the variables $p_1 \dots, p_n$.

One can then ask, which Boolean functions can be expressed in propositional logic. In fact, propositional logic is expressively complete (in the standard Tarskian setting). It is not difficult to write the definition of any Boolean function in propositional logic by simply writing out a big disjunction of all the combinations of assignments to variables that are mapped to true by the Boolean function.

Proposition 14. *Every Boolean function can be defined in propositional logic.*

How can we generalise this definition and result to logics with team semantics? Remember that the satisfying element in team semantics is a set of assignments, and that will then need to be reflected in any meaningful experssivity characterisation result.

How to characterise expressivity – Team semantics – definitions

In team semantics setting, a propositional formula defines a set of teams that satisfy it.

Definition 15. We define

$$\mathrm{Teams}(\varphi) \coloneqq \{T \mid T \models \varphi\}$$

We then want to know, what are the families of teams that can be written as $\mathrm{Teams}(\varphi)$ by some formula φ.
Definitions of downward/union closure and flatness generalise to families of teams.

Definition 16. A family of teams $\mathcal{T}$ is

- downward closed, if ($T \in \mathcal{T}$ and $S \subseteq T$) implies $S \in \mathcal{T}$.
- union closed, if $T, S \in \mathcal{T}$ implies $T \cup S \in \mathcal{T}$.
- flat, if $T \in \mathcal{T}$ if and only if $\{t\} \in \mathcal{T}$, for all $t \in T$.

These properties, together with the empty team property constitute the main closure properties of team-based logics, and are in fact closely connected to each other, as can be seen from the following results.

Properties of families of teams

Proposition 17. *A family of teams $\mathcal{T}$ is flat if and only if it is union & downward closed and $\emptyset \in \mathcal{T}$.*

Proof. Left-to-right direction if trivial. For the right-to-left direction, assume that $\mathcal{T}$ is union & downward closed and that $\emptyset \in \mathcal{T}$. Now the left-to-right direction of

$$T \in \mathcal{T} \iff \forall t \in T : \{t\} \in \mathcal{T}$$

follows from downward closure, while the converse direction follows from union closure. The empty team property is required to omit the special case of $\mathcal{T} = \emptyset$. □

Proposition 18. *Let $\mathcal{T}$ be a flat family of teams. Then $T \in \mathcal{T}$ if and only if $T \subseteq \bigcup \mathcal{T}$.*

Proof. Left-to-right direction is trivial and follows directly from the definition of a union.
Right-to-left direction: By Proposition 17, $\mathcal{T}$ is union closed and downward closed. From union closure of $\mathcal{T}$ it follows that $\bigcup \mathcal{T} \in \mathcal{T}$. Now since $\mathcal{T}$ is downward closed and $T \subseteq \bigcup \mathcal{T}$, if follows that $T \in \mathcal{T}$. □

How to characterise expressivity – Team semantics – results

We have already seen (and partly proved) the following closure results:

Proposition 19.
- *A family of teams defined by a PL-formula is flat.*
- PL[dep]*-definable team families are downward closed and include the empty team.*

Proof. Flatness is proven by structural induction. The cases for atomic formulae follow directly from their semantics. The case for $\wedge$ is trivial. Assume flatness holds for φ and ψ.

$$T \models \varphi \vee \psi \iff T_1 \models \varphi \text{ and } T_2 \models \psi \text{ for some } T_1 \cup T_2 = T$$

By IH, the right-hand side is equivalent to: $\forall t \in T : \{t\} \models \varphi$ or $\{t\} \models \psi$. This is again equivalent to $\forall t \in T : \{t\} \models \varphi \vee \psi$, due to the empty team property. □

Interestingly the above results can be strengthened to if and only if!

We will first show that every flat family of teams is definable in propositional logic. From the fact that PL is flat, our first precise characterisation follows.

Expressivity of PL with team semantics

Proposition 20. *For every flat family $\mathcal{T}$ there exists a PL-formula φ such that $\mathcal{T} = \mathrm{Teams}(\varphi)$.*

Proof. Let $\mathcal{T}$ be a flat family of teams using proposition symbols $p_1, \ldots, p_n$. For every assignment s over the propositions $p_1, \ldots, p_n$, let φ_s be a PL-formula whose only satisfying assignment is s. This exists by Proposition 14. We will then define

$$\Phi \coloneqq \bigvee_{s \in \bigcup \mathcal{T}} \varphi_s$$

and claim that $\mathcal{T} = \mathrm{Teams}(\Phi)$. It is easy to check that $T \models \Phi$ if and only if $T \subseteq \bigcup \mathcal{T}$. By Proposition 18, the latter holds if and only if $T \in \mathcal{T}$. □

We have proven our first characterisation result:

Theorem 21. *A family of teams is definable in PL if and only if the family is flat.*

We will next turn towards the characterisation of the expressivity of PL[dep]. We have already shown that the logic is downward closed and has the empty team property. Before proving the converse direction of the characterisation, we will consider a simpler logic that turns out to be equi-expressive with PL[dep].

Expressivity: the downward closed case

Let's consider an extension PL[$\otimes$] of PL with the so-called Boolean disjunction

$$T \models \varphi \otimes \psi \text{ if and only if } T \models \varphi \text{ or } T \models \psi.$$

The following proposition can be proven by structural induction. We only show the case for $\otimes$ for downward closure.

Proposition 22. PL[$\otimes$] *is downward closed and has the empty team property.*

Proof. Let $S \subseteq T$ be teams such that $T \models \varphi \otimes \psi$ and that φ and ψ define downward closed properties. Then, by the semantics of $\otimes$, $T \models \varphi$ or $T \models \varphi$. By induction hypothesis $S \models \varphi$ or $S \models \varphi$. And finally, by the semantics of $\otimes$, $S \models \varphi \otimes \psi$. □

It is easy to note that dependence atoms can be expressed in PL[$\otimes$]:

$$T \models \mathrm{dep}(p_1, \ldots, p_n, q) \text{ if and only if } T \models \bigvee_{b \in \{\bot, \top\}^n} \big(p_1^{b_1} \wedge \cdots \wedge p_n^{b_n} \wedge (q \otimes \neg q)\big),$$

where $p^{\bot} \coloneqq \neg p$ and $p^{\top} \coloneqq p$. In order to realise that the above translation is indeed correct, note first that a team T satisfies $q \otimes \neg q$, if and only if the truth value of q is constant in T. That is, every assignment in T gives the same truth value for q. Now

$$T \models \bigvee_{b \in \{\bot, \top\}^n} \big(p_1^{b_1} \wedge \cdots \wedge p_n^{b_n} \wedge (q \otimes \neg q)\big)$$

if and only if we can split T into 2^n many pieces T_b, $b \in \{\bot, \top\}^n$, such that each piece

$$T_b \models \big(p_1^{b_1} \wedge \cdots \wedge p_n^{b_n} \wedge (q \mathbin{⫽} \neg q)\big).$$

Hence, we are saying that T can be split into 2^n pieces according to the truth values given to $p_1, \ldots, p_n$, and in each of those pieces the truth value of q is constant. This is equivalent to saying that $p_1, \ldots, p_n$ functionally determine q, that is $T \models \mathrm{dep}(p_1, \ldots, p_n, q)$.

We are now ready to formalise our second characterisation result.

Expressive power of PL[⫽]

Theorem 23. *A family of teams is definable in* PL[⫽] *if and only if the family is downward closed and includes the empty team.*

Proof. We need to show that every downward closed family of teams that includes the empty team is definable in PL[⫽] as the converse direction follows from Proposition 22. Let $\mathcal{T}$ be a family of teams with the aforementioned properties. Define

$$\Phi \coloneqq \bigdiv_{T \in \mathcal{T}} \bigvee_{s \in T} \varphi_s, \text{ where } \varphi_s \text{ is a formula whose only satisfying assignment is } s.$$

We claim that $\mathrm{Teams}(\Phi) = \mathcal{T}$. If $S \models \Phi$, there is some $T \in \mathcal{T}$ s.t. $S \models \bigvee_{s \in T} \varphi_s$. Thus $S \subseteq T$ and hence $S \in \mathcal{T}$, for $\mathcal{T}$ is downward closed. Conversely, if $S \in \mathcal{T}$ then $S \models \bigvee_{s \in S} \varphi_s$, and thus $S \models \Phi$. □

The formula Φ can be made smaller by considering only subset maximal teams in $\mathcal{T}$ in the definition of Φ.

We have already shown that every PL[dep]-formula can be translated to an equivalent PL[⫽]-formula with an exponential blow-up in the size of the formula. Interestingly, the converse direction holds as well. We will next show that the expressive power of PL[dep] and PL[⫽] coincide, and that exponential blow-up is sometimes necessary when PL[dep]-formulae are translated to PL[⫽].

Types and characterising formulae

We define some auxiliary notation and formulae:

- $\mathrm{Type}_\Psi(s) \coloneqq \{\varphi \in \Psi \mid s \models \varphi\}$, for a set of PL-formulae Ψ and an assignment s.
- For $\Gamma \subseteq \Psi$, define $\theta_\Gamma \coloneqq \bigwedge_{\psi \in \Gamma} \psi \wedge \bigwedge_{\psi \in \Psi \setminus \Gamma} \neg\psi$, where the negation in $\neg\psi$ is pushed to the atomic level.

It is easy to check that $\mathrm{Type}_\Psi(s) = \Gamma$ if and only if $s \models \theta_\Gamma$.

- $\mathrm{Type}_\Psi(T) \coloneqq \{\mathrm{Type}_\Psi(s) \mid s \in T\}$, for a team T.

Lemma 24. *Assume that T and S be teams and let Ψ be a finite set of PL-formulae.*

1. *For each $\psi \in \Psi$, $T \models \psi$ if and only if $\psi \in \bigcap \mathrm{Type}_\Psi(T)$.*

2. If $T \models \Circledown \Psi$ *and* $\mathrm{Type}_\Psi(S) \subseteq \mathrm{Type}_\Psi(T)$*, then* $S \models \Circledown \Psi$*.*

Proof. (1.) If $T \models \psi$, then by Proposition 19 (flatness), $s \models \psi$ for every $s \in T$, which means that $\psi \in \mathrm{Type}_\Psi(s)$ for every $s \in T$. On the other hand, if $\psi \in \bigcap \mathrm{Type}_\Psi(T)$, then $s \models \psi$ for every $s \in T$. By Proposition 19 (flatness), it follows that $T \models \psi$.

(2.) Assume that $T \models \Circledown \Psi$ and $\mathrm{Type}_\Psi(S) \subseteq \mathrm{Type}_\Psi(T)$. Thus, $T \models \psi$ for some $\psi \in \Psi$, and by claim (1.), $\psi \in \bigcap \mathrm{Type}_\Psi(T)$. Since $\mathrm{Type}_\Psi(S) \subseteq \mathrm{Type}_\Psi(T)$, it follows that $\psi \in \bigcap \mathrm{Type}_\Psi(S)$. Thus, $S \models \psi$, and consequently $S \models \Circledown \Psi$. □

Expressive power of PL[dep]

Consider next the formula stating that the truth value w.r.t. a set of propositions $\Psi \subseteq \mathsf{PROP}$ is constant:

$$\gamma \coloneqq \bigwedge_{p \in \Psi} \mathrm{dep}(p).$$

Hence $T \models \gamma$ if and only if $|\mathrm{Type}_\Psi(T)| \leq 1$. Define now recursively

$$\gamma^0 \coloneqq p \wedge \neg p, \qquad \gamma^{k+1} \coloneqq (\gamma^k \vee \gamma).$$

It is easy to show by induction that $T \models \gamma^k$ if and only if $|\mathrm{Type}_\Psi(T)| \leq k$.

Lemma 25. *If* $\Psi \subseteq \mathsf{PROP}$ *is a finite set of propositions and* $T \neq \emptyset$ *a team, there is a* $\xi_T \in \mathrm{PL[dep]}$ *s.t. for every* S

$$S \models \xi_T \quad \iff \quad \mathrm{Type}_\Psi(T) \not\subseteq \mathrm{Type}_\Psi(S).$$

Proof. Let $|\mathrm{Type}_\Psi(T)| = k+1$. Recall θ_Γ is a characterisic formula of Γ. We define

$$\xi_T \coloneqq \Big(\bigvee_{\Gamma \in X} \theta_\Gamma \Big) \vee \gamma^k, \text{ where } X = \mathcal{P}(\Psi) \setminus \mathrm{Type}_\Psi(T).$$

Now given a team S we have

$$\begin{aligned} S \models \xi_T &\iff \text{there are } T_1, T_2 \text{ s.t. } T_1 \cup T_2 = S, \mathrm{Type}_\Psi(T_1) \subseteq X, |\mathrm{Type}_\Psi(T_2)| \leq k \\ &\iff |\mathrm{Type}_\Psi(T) \cap \mathrm{Type}_\Psi(S)| \leq k \\ &\iff \mathrm{Type}_\Psi(T) \not\subseteq \mathrm{Type}_\Psi(S). \quad \square \end{aligned}$$

We have already shown that every PL[dep]-formula can be translated to an equivalent PL[⍰]-formula. We are now ready to show the converse.

Theorem 26. PL[⍰] *is equi-expressive with* PL[dep]*.*

Proof. PL[⍰] ≤ PL[dep] direction: Let $\varphi = \Circledown \Psi$ be a PL[⍰]-formula in a normal form, where $\Psi \subseteq \mathrm{PL}$. Define

$$\eta \coloneqq \bigwedge_{T \not\in \mathrm{Teams}(\varphi)} \xi_T, \text{ where } \xi_T \text{ is as in Lemma 25.}$$

Intuitively $S \models \eta$ iff no falsifying team of φ is completely subsumed by S. Above, we consider teams over proposition symbols that occur in φ, which makes η finite.
By definition η is a PL[dep]-formula. To prove that $\mathrm{Teams}(\eta) = \mathrm{Teams}(\varphi)$, assume first that $S \in \mathrm{Teams}(\varphi)$, and consider any $T \notin \mathrm{Teams}(\varphi)$. It follows from Lemma 24 that $\mathrm{Type}_\Psi(T) \not\subseteq \mathrm{Type}_\Psi(S)$. Hence by Lemma 25, $S \models \xi_T$. Thus $S \in \mathrm{Teams}(\eta)$.

Assume then that $S \notin \mathrm{Teams}(\varphi)$. Since $\mathrm{Type}_\Psi(S) \subseteq \mathrm{Type}_\Psi(S)$, it follows from Lemma 25 that $S \not\models \xi_S$. Thus $S \notin \mathrm{Teams}(\eta)$. □

We have now showed that PL[⍰] and PL[dep] both capture the team families that are downward closed and include the empty team. What remains is to show that the exponential blow-up in the translation from PL[dep] to PL[⍰] is inevitable.

Dimensions of team families

Definition 27. The *lower dimension* $\dim(\varphi)$ of a formula φ to is the least n such that

$$T \models \varphi \iff S \models \varphi \text{ for all } S \subseteq T \text{ s.t. } |S| \leq n.$$

The lower dimension of a flat formula is 1, and for a dependence atom it is 2. The lower dimension is not easy to approximate compositionally, for that we define the notion of upper dimension. Define $M(\varphi)$ as the set of subset maximal teams satisfying φ.

Definition 28. The *upper dimension* $\mathrm{Dim}(\varphi)$ of a formula φ is the cardinality of $M(\varphi)$.

Interestingly, $\mathrm{Dim}(\varphi)$ can be given sharp compositional estimates, and it can be shown that $\dim(\varphi) \leq \mathrm{Dim}(\varphi)$. We omit the proof of $\dim(\varphi) \leq \mathrm{Dim}(\varphi)$, but it can be found in [Hel+14].

Next we present the compositional estimates for $\mathrm{Dim}(\varphi)$.

Estimates for the upper dimension

Lemma 29. *We have the following upper dimension estimates for $\varphi, \psi \in \mathrm{PL}[⍰]$:*

1. $\mathrm{Dim}(p) = \mathrm{Dim}(\neg p) = 1$.
2. $\mathrm{Dim}(\varphi \wedge \psi) \leq \mathrm{Dim}(\varphi)\,\mathrm{Dim}(\psi)$.
3. $\mathrm{Dim}(\varphi \vee \psi) \leq \mathrm{Dim}(\varphi)\,\mathrm{Dim}(\psi)$.
4. $\mathrm{Dim}(\varphi \mathbin{⍰} \psi) \leq \mathrm{Dim}(\varphi) + \mathrm{Dim}(\psi)$.

Proof. We omit the cases for (1) and (3), since (1) is trivial, and (3) is analogous to (2).
Case (2): Note that $\mathrm{Teams}(\varphi \wedge \psi) = \mathrm{Teams}(\varphi) \cap \mathrm{Teams}(\psi)$. By IH, $\mathrm{Teams}(\varphi)$ and $\mathrm{Teams}(\psi)$ are finitely generated by $M(\varphi)$ and $M(\psi)$, respectively. Moreover, $|M(\varphi)| \leq \mathrm{Dim}(\varphi)$ and $|M(\psi)| \leq \mathrm{Dim}(\psi)$. It is immediate that

$$M(\varphi \wedge \psi) \subseteq \{T \cap U \mid T \in M(\varphi), U \in M(\psi)\}.$$

Clearly, by the IH the right-hand side of the inclusion above also generates the family $\mathrm{Teams}(\varphi \wedge \psi)$. The inclusion now implies $|M(\varphi \wedge \psi)| \leq |M(\varphi) \times M(\psi)| \leq \mathrm{Dim}(\varphi)\,\mathrm{Dim}(\psi)$. Hence, $\mathrm{Dim}(\varphi \wedge \psi) \leq \mathrm{Dim}(\varphi)\,\mathrm{Dim}(\psi)$.
Case (4): For the Boolean disjunction, it holds that

$$M(\varphi \mathbin{⩒} \psi) \subseteq M(\varphi) \cup M(\psi)$$

and the right-hand side of the inclusion generates the family $\mathrm{Teams}(\varphi \mathbin{⩒} \psi)$. The dimension estimate follows immediately. □

Next we will show, how the notion of a dimension can be used to prove that dependence atoms cannot have short definitions in PL[⩒].

What are dimensions good for?

Proposition 30. $\mathrm{Dim}(\mathrm{dep}(p_1, \ldots, p_n, q)) = 2^{2^n}$.

Proposition 31. *For $\varphi \in \mathrm{PL}[⩒]$, $\mathrm{Dim}(\varphi) \leq 2^k$, where k is the number of occurrences of ⩒ in φ.*

Theorem 32. *Let $\varphi \in \mathrm{PL}[⩒]$ such that $\mathrm{Teams}(\varphi) = \mathrm{Teams}(\mathrm{dep}(p_1, \ldots, p_n, q))$. Then φ contains more than 2^n symbols.*

Proof. By Prop 30, $\mathrm{Dim}(\varphi) = \mathrm{Dim}(\mathrm{dep}(p_1, \ldots, p_n, q)) = 2^{2^n}$. Thus $2^{2^n} \leq 2^{\mathrm{occ}_{⩒}(\varphi)}$ by Prop. 31, implying $2^n \leq \mathrm{occ}_{⩒}(\varphi)$. Hence φ has at least 2^n Boolean disjunctions. □

Thus, any translation from PL[dep] to PL[⩒] leads to an exponential blow-up.

In addition to characterisations of expressivity in the downward closed setting similar results exist for union closed logics.

Expressivity of propositional inclusion logic

Theorem 33. *A family of teams is definable in $\mathrm{PL}[\subseteq]$ if and only if it is union closed and includes the empty team.*

Proof. We will omit the proof, which combines ideas from the characterisation of PL[⩒] and its equivalence with PL[dep]. The result was first shown in [HS15]. □

Conclusion of Lecture 2

- Properties of families of teams.
- Expressivity characterisation of PL[⩒].
- Equivalence of PL[⩒] and PL[dep].
- Expressivity characterisation of $\mathrm{PL}[\subseteq]$.

3 Inclusion Logic

Literature: [Hel+19; Hel+20]

Interestingly to note, PL[⊆] definable classes of propositional teams are exactly those $\mathcal{C}$ such that

- $\emptyset \in \mathcal{C}$ and
- $\mathcal{C}$ is union closed ($X \in \mathcal{C}, Y \in \mathcal{C} \Rightarrow X \cup Y \in \mathcal{C}$).

The latter we will prove in the following.

Inclusion

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 34. PL[⊆] *is union closed.*

Proof. Let $\varphi \in$ PL[⊆] and $T_1 \models \varphi$ and $T_2 \models \varphi$ and $T_1 \neq T_2$. We do an induction on the structure of φ.

- $\varphi = p$: $T_1 \models p$ and $T_2 \models p$ imply $T_1 \cup T_2 \models p$.
- $\varphi = \bar{p} \subseteq \bar{q}$: as $T_1 \models \bar{p} \subseteq \bar{q}$ and $T_2 \models \bar{p} \subseteq \bar{q}$, we have in each case that for all $u \in T_1$ there is a $v \in T_1$ such that $u(\bar{p}) = v(\bar{q})$ and for all $u \in T_2$ there is a $v \in T_2$ such that $u(\bar{p}) = v(\bar{q})$. As a result, for all $u \in T_1 \cup T_2$ there is a $v \in T_1 \cup T_2$ such that $u(\bar{p}) = v(\bar{q})$. This means that $T_1 \cup T_2 \models \bar{p} \subseteq \bar{q}$.
- $\varphi = \alpha \wedge \beta$: by IH, $T_1 \cup T_2 \models \alpha$ and $T_1 \cup T_2 \models \beta$. Thus $T_1 \cup T_2 \models \alpha \wedge \beta$.
- $\varphi = \alpha \vee \beta$: by IH, $T_1 \cup T_2 \models \alpha$ and $T_1 \cup T_2 \models \beta$. Thus $T_1 \cup T_2 \models \alpha \vee \beta$.

□

The union closure property allows us to give a simple proof that the validity problem for PL[⊆] is coNP-complete.

Validity in Team Semantics

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

Validity in Inclusion Logic is Hard

Theorem 35. VAL(PL[⊆]) *is* coNP*-complete.*

Proof. Hardness: VAL(PL) is coNP-complete.
Membership:

1. PL[⊆] is union closed.
2. $\varphi \in \mathrm{PL}[\subseteq]$ is valid iff φ is valid on singleton teams.
3. On singleton teams inclusion atoms can be eliminated.

$$p_1, \dots, p_n \subseteq q_1, \dots, q_n \text{ can be rewritten as } (p_1 \leftrightarrow q_1) \wedge \dots \wedge (p_n \leftrightarrow q_n)$$

4. Check validity of the PL-translatee.

□

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 $b_1, \dots, b_n \in \{0, 1\}$
Question: is the output of the circuit 1

Proposition 36 ([Gol77])**.** MCVP *is* P*-complete w.r.t.* $\leq_m^{\log}$*-reductions.*

Model-Checking for Inclusion Logic

Theorem 37 ([Hel+19, Thm. 3.5])**.** PL[⊆]-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 PL[⊆]-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

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

- $s_\bot(p) = 1$ 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 PL[⊆]-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:

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

Encoding MCVP: the final puzzle pieces
More truth about the team:

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

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.
Crux 2: $\neg p_\bot$ and $s_\bot$ ensure that Y is nonempty and deal with the propagation of the value 0 by the subformulae of the form $p_i \subseteq p_j$.

In the following, we present a full formal proof.

Proof. We will establish a $\leq_m^{\log}$-reduction from MCVP to the model checking problem of PL[$\subseteq$] under lax semantics. Since MCVP is P-complete, the claim follows. More precisely, we will show how to construct, for each monotone Boolean circuit C with n input gates and for each input $\vec{b}$ for C, a team $X_{C,\vec{b}}$ and a PL[$\subseteq$]-formula φ_C such that

$$X_{C,\vec{b}} \models \varphi_C \text{ if and only if the output of the circuit } C \text{ with the input } \vec{b} \text{ is 1.}$$

We use teams to encode valuations of the circuit. For each gate v_i of a given circuit, we associate an assignment s_i. The crude idea is that if s_i is in the team under consideration, the value of the gate v_i with respect to the given input is 1. The formula φ_C is used to quantify a truth value for each Boolean gate of the circuit, and then for checking that the truth values of the gates propagate correctly. We next define the construction formally.

Let $C = (V, E, \alpha)$ be a monotone Boolean circuit with n input gates and one output gate and let $\vec{b} = (b_1 \dots b_n) \in \{0,1\}^n$ be an input to the circuit C. We stipulate that $V = \{v_0, \dots, v_m\}$ and that v_0 is the output gate of C. Define

$$\tau_C \coloneqq \{p_0, \dots, p_m, p_\top, p_\bot\} \cup \{p_{k=i\vee j} \mid i < j, \alpha(v_k) = \vee, \text{ and } (v_i, v_k), (v_j, v_k) \in E\}.$$

For each $i \leq m$, we define the assignment $s_i \colon \tau_C \to \{0,1\}$ as follows:

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

Furthermore, we define $s_\bot(p) = 1$ iff $p = p_\bot$ or $p = p_\top$. We note that the assignment $s_\bot$ will be the only assignment that maps $p_\bot$ to 1. We make use of the fact that for each gate v_i of C, we have that $s_\bot(p_i) = 0$. We define

$$X_{C,\vec{b}} \coloneqq \big\{s_i \mid \alpha(v_i) \in \{\wedge, \vee\}\big\} \cup \big\{s_i \mid \alpha(v_i) \in \{x_i \mid b_i = 1\}\big\} \cup \{s_\bot\},$$

that is, $X_{C,\vec{b}}$ consists of assignments for each of the Boolean gates, assignments for those input gates that are given 1 as an input, and of the auxiliary assignment $s_\bot$.

Let X be any nonempty subteam of $X_{C,\vec{b}}$ such that $s_\bot \in X$. We have

$$\begin{aligned} X &\models p_\top \subseteq p_0 && \text{iff } s_0 \in X \\ X &\models p_i \subseteq p_j && \text{iff } (s_i \in X \text{ implies } s_j \in X) \\ X &\models p_k \subseteq p_{k=i\vee j} && \text{iff } (i < j, (v_i, v_k), (v_j, v_k) \in E, \alpha(v_k) = \vee \\ & && \quad \text{and } s_k \in X \text{ implies that } s_i \in X \text{ or } s_j \in X). \end{aligned} \tag{3.1}$$

Recall the intuition that $s_i \in X$ should hold iff the value of the gate v_i is 1. Define

$$\begin{aligned} \psi_{\text{out}=1} &\coloneqq p_\top \subseteq p_0, \\ \psi_\wedge &\coloneqq \bigwedge\{p_i \subseteq p_j \mid (v_j, v_i) \in E \text{ and } \alpha(p_i) = \wedge\}, \\ \psi_\vee &\coloneqq \bigwedge\{p_k \subseteq p_{k=i\vee j} \mid i < j, (v_i, v_k) \in E, (v_j, v_k) \in E, \text{ and } \alpha(v_k) = \vee\}, \\ \varphi_C &\coloneqq \neg p_\bot \vee (\psi_{\text{out}=1} \wedge \psi_\wedge \wedge \psi_\vee). \end{aligned}$$

We claim that $X_{C,\vec{b}} \models \varphi_C$ iff the output of C with the input $\vec{b}$ is 1.

The idea of the reduction is the following: The disjunction in ϕ_C is used to guess a team Y for the right disjunct that encodes the valuation β of the circuit C. The right disjunct is then evaluated with respect to the team Y with the intended meaning that $\beta(v_i) = 1$ whenever $s_i \in Y$. Note that Y is always as required in (3.1). The formula $\psi_{\text{out}=1}$ is used to state that $\beta(v_0) = 1$, whereas the formulae $\psi_\wedge$ and $\psi_\vee$ are used to propagate the truth value 1 down the circuit. The assignment $s_\bot$ and the proposition $p_\bot$ are used to make sure that Y is nonempty and to deal with the propagation of the value 0 by the subformulae of the form $p_i \subseteq p_j$.

Now observe that the team $X_{C,\vec{b}}$ can be easily computed by a logspace Turing machine which scans the input for $\wedge$-gates, $\vee$-gates, and true input gates, and then outputs the corresponding team members s_i in a bitwise fashion. The formula φ_C can be computed in logspace as well:

1. the left disjunct does not depend on the input,
2. for $\psi_\wedge$ we only need to scan for the $\wedge$-gates and output the inclusion-formulae for the corresponding edges,
3. for $\psi_\vee$ we need to maintain two binary counters for i and j, and use them for searching for those disjunction gates that satisfy $i < j$.

Consequently, the reduction can be computed in logspace. □

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 PL[$\subseteq$] ensures that this always exists.

Lemma 38 (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). For literals (propositions or their negations) the claim follows via flatness.

P-algorithm for PL[$\subseteq$]-MC
Important properties:

- Each team T has a unique maximal subteam satisfying a given formula φ.
- For literals maxsub(T, φ) is computable in polynomial time (Lemma 38).

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:
   a) For literals φ labelled by Y, replace Y by $\mathrm{maxsub}(Y, \varphi)$.
   b) 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:
   a) 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

bottum-up part (odd i):

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

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 $f_i(\gamma) \coloneqq f_{i-1}(\gamma) \cap f_i(\theta \vee \gamma)$.

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

Proof. We will present a labelling algorithm for model checking $T \models \varphi$. Let $\mathrm{subOcc}(\varphi)$ denote the set of all *occurrences* of subformulae of φ. Below we denote occurrences as if they were formulae, but we actually refer to some particular occurrence of the formula.

A function $f\colon \mathrm{subOcc}(\varphi) \to \mathcal{P}(T)$ is called a labelling function of φ. We will next give an algorithm for computing a sequence $f_0, f_1, f_2, \ldots$, of such labelling functions.

- Define $f_0(\psi) = W$ for each $\psi \in \mathrm{subOcc}(\varphi)$.
- For odd $i \in \mathbb{N}$, define $f_i(\psi)$ bottom up as follows:
  1. For literal ψ, define $f_i(\psi) \coloneqq \mathrm{maxsub}(f_{i-1}(\psi), \psi)$.
  2. $f_i(\psi \wedge \theta) \coloneqq f_i(\psi) \cap f_i(\theta)$.
  3. $f_i(\psi \vee \theta) \coloneqq f_i(\psi) \cup f_i(\theta)$.
- For even $i \in \mathbb{N}$ larger than 0, define $f_i(\psi)$ top to bottom as follows:

1. Define $f_i(\varphi) \coloneqq f_{i-1}(\varphi) \cap T$.
2. If $\psi = \theta \wedge \gamma$, define $f_i(\theta) \coloneqq f_i(\gamma) \coloneqq f_i(\theta \wedge \gamma)$.
3. If $\psi = \theta \vee \gamma$, define $f_i(\theta) \coloneqq f_{i-1}(\theta) \cap f_i(\theta \vee \gamma)$ and $f_i(\gamma) \coloneqq f_{i-1}(\gamma) \cap f_i(\theta \vee \gamma)$.

By a straightforward induction on i, we can prove that $f_{i+1}(\psi) \subseteq f_i(\psi)$ for every $\psi \in \mathrm{subOcc}(\varphi)$. The only nontrivial induction step is that for $f_{i+1}(\theta)$ and $f_{i+1}(\gamma)$, when $i+1$ is even and $\psi = \theta \wedge \gamma$. To deal with this step, observe that, by the definition of f_{i+1} and f_i, we have $f_{i+1}(\theta) = f_{i+1}(\gamma) = f_{i+1}(\psi)$ and $f_i(\psi) \subseteq f_i(\theta), f_i(\gamma)$. Note also that for even $i+1$ the direction of the proof is from larger formulae to subformulae; in particular we may assume that $f_{i+1}(\psi) \subseteq f_i(\psi)$. Now by connecting the previous observations, we obtain that $f_{i+1}(\theta) = f_{i+1}(\psi) \subseteq f_i(\psi) \subseteq f_i(\theta)$ and $f_{i+1}(\gamma) = f_{i+1}(\psi) \subseteq f_i(\psi) \subseteq f_i(\gamma)$.

It follows that there is an integer $j \leq 2 \cdot |W| \cdot |\varphi|$ such that $f_{j+2} = f_{j+1} = f_j$. We denote this fixed point f_j of the sequence $f_0, f_1, f_2, \ldots$ by f_∞. By Lemma 38 the outcome of $\mathrm{maxsub}(\cdot,\cdot)$ is computable in polynomial time with respect to its input. That being, clearly f_{i+1} can be computed from f_i in polynomial time with respect to $|W| + |\varphi|$. On that account f_∞ is also computable in polynomial time with respect to $|W| + |\varphi|$.

We will next prove by induction on $\psi \in \mathrm{subOcc}(\varphi)$ that $f_\infty(\psi) \models \psi$. Note first that there is an odd integer i and an even integer j such that $f_\infty = f_i = f_j$.

1. If ψ is a literal, the claim is true since $f_\infty = f_i$ and $f_i(\psi) = \mathrm{maxsub}(f_{i-1}(\psi), \psi)$.
2. Assume next that $\psi = \theta \wedge \gamma$, and the claim is true for θ and γ. Since $f_\infty = f_j$, we have $f_\infty(\psi) = f_\infty(\theta) = f_\infty(\gamma)$, as a result, by induction hypothesis, $f_\infty(\psi) \models \theta \wedge \gamma$.
3. In the case $\psi = \theta \vee \gamma$, we obtain the claim $f_\infty(\psi) \models \psi$ by using the induction hypothesis, and the observation that $f_\infty(\psi) = f_i(\psi) = f_i(\theta) \cup f_i(\gamma) = f_\infty(\theta) \cup f_\infty(\gamma)$.

In particular, if $f_\infty(\varphi) = T$, then $T \models \varphi$. Consequently, to complete the proof of the lemma, it suffices to prove that the converse implication is true, as well. To prove this, assume that $T \models \varphi$. Then for each $\psi \in \mathrm{subOcc}(\varphi)$, there is a team T_ψ such that

1. $T_\varphi = T$.
2. If $\psi = \theta \wedge \gamma$, then $T_\psi = T_\theta = T_\gamma$.
3. If $\psi = \theta \vee \gamma$, then $T_\psi = T_\theta \cup T_\gamma$.
4. If ψ is a literal, then $T_\psi \models \psi$.

We prove by induction on i that $T_\psi \subseteq f_i(\psi)$ for all $\psi \in \mathrm{subOcc}(\varphi)$. For $i = 0$, this is obvious, since $f_0(\psi) = W$ for all ψ. Assume next that $i+1$ is odd and the claim is true for i. We prove the claim $T_\psi \subseteq f_i(\psi)$ by induction on ψ.

1. If ψ is a literal, then $f_{i+1}(\psi) = \mathrm{maxsub}(f_i(\psi), \psi)$. Since $T_\psi \models \psi$, and by induction hypothesis, $T_\psi \subseteq f_i(\psi)$, the claim $T_\psi \subseteq f_{i+1}(\psi)$ is true.

2. Assume that $\psi = \theta \wedge \gamma$. By induction hypothesis on θ and γ, we have $T_\psi = T_\theta \subseteq f_{i+1}(\theta)$ and $T_\psi = T_\gamma \subseteq f_{i+1}(\gamma)$. For this reason, we get $T_\psi \subseteq f_{i+1}(\theta) \cap f_{i+1}(\gamma) = f_{i+1}(\psi)$.

3. The case $\psi = \theta \vee \gamma$ is similar to the previous one; we omit the details.

Assume then that $i+1$ is even and the claim is true for i. This time we prove the claim $T_\psi \subseteq f_i(\psi)$ by top to bottom induction on ψ.

1. By assumption, $T_\varphi = T$, whence by induction hypothesis, $T_\varphi \subseteq f_i(\varphi) \cap T = f_{i+1}(\varphi)$.

2. Assume that $\psi = \theta \wedge \gamma$. By induction hypothesis on ψ, we have $T_\psi \subseteq f_{i+1}(\psi)$. Since $T_\psi = T_\theta = T_\gamma$ and $f_{i+1}(\psi) = f_{i+1}(\theta) = f_{i+1}(\gamma)$, this implies that $T_\theta \subseteq f_{i+1}(\theta)$ and $T_\gamma \subseteq f_{i+1}(\gamma)$.

3. Assume that $\psi = \theta \vee \gamma$. Using the fact that $T_\theta \subseteq T_\psi$, and the two induction hypotheses on i and ψ, we see that $T_\theta \subseteq f_i(\theta) \cap T_\psi \subseteq f_i(\theta) \cap f_{i+1}(\psi) = f_{i+1}(\theta)$. Similarly, we see that $T_\gamma \subseteq f_{i+1}(\gamma)$.

It follows now that $T = T_\varphi \subseteq f_\infty(\varphi)$. Since $f_\infty(\varphi) \subseteq f_2(\varphi) \subseteq T$, we conclude that $f_\infty(\varphi) = T$. This completes the proof of the implication $T \models \varphi \;\Rightarrow\; f_\infty(\varphi) = T$.

□

Satisfiability for Inclusion Logic

Theorem 39 ([Hel+20, Cor. 3.6])**.** PL[⊆]-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 Path-Systems variant *(our focus)*

The upper bound result is established by an equivalence-preserving translation to *propositional dynamic logic* extended with the global and converse modalities. It is well-known that this logic is complete for EXP [BRV01; Hem96; Eij14]. In fact, we will only need multimodal logic with the global modality for our purposes.

Let Π and $\mathcal{R}$ be countably infinite sets of proposition symbols and binary relation symbols, respectively. The following grammar defines a modal language $\mathcal{L}$:

$$\varphi ::= p \mid \neg\varphi \mid (\varphi_1 \wedge \varphi_2) \mid \langle R\rangle\varphi \mid \langle E\rangle\varphi,$$

where $p \in \Pi$, $R \in \mathcal{R}$, and E is a novel symbol. The (classical Kripke-styled) semantics of $\mathcal{L}$ is defined with respect to ordinary pointed Kripke models (M, w) for multimodal logic.

Let $M = (W, \{R\}_{R\in\mathcal{R}}, V)$ be a Kripke model, where $V\colon \Pi \to \mathcal{P}(W)$ is the *valuation* function interpreting proposition symbols. Let $w \in W$. The following clauses define

the semantics of $\mathcal{L}$; note that we use the turnstile $\Vdash$ instead of $\models$, which is reserved for team semantics in this lecture.

$$
\begin{aligned}
M, w \Vdash p &\Leftrightarrow w \in V(p)\\
M, w \Vdash \neg\varphi &\Leftrightarrow M, w \nVdash \varphi\\
M, w \Vdash \varphi_1 \wedge \varphi_2 &\Leftrightarrow M, w \Vdash \varphi_1 \text{ and } M, w \Vdash \varphi_2\\
M, w \Vdash \langle R\rangle\varphi &\Leftrightarrow M, u \Vdash \varphi \text{ for some } u \text{ such that } wRu\\
M, w \Vdash \langle E\rangle\varphi &\Leftrightarrow M, u \Vdash \varphi \text{ for some } u \in W
\end{aligned}
$$

We next define a satisfiability-preserving translation from propositional inclusion logic into $\mathcal{L}$. We let $[R]$ and $[E]$ denote $\neg\langle R\rangle\neg$ and $\neg\langle E\rangle\neg$, respectively. Before we fix the translation, we define some auxiliary formulae. Let θ be a formula of PL[$\subseteq$]. We let $\mathbf{SUB}(\theta)$ denote the set of subformulae of θ; we distinguish all instances of subformulae, so for example $p \wedge p$ has *three* subformulae (the right and the left instances of p and the conjunction itself).

For each formula $\varphi \in \mathbf{SUB}(\theta)$, fix a fresh proposition symbol p_φ that does not occur in θ. We next define, for each $\varphi \in \mathbf{SUB}(\theta)$, a novel auxiliary formula χ_φ. If $\varphi \in \mathbf{SUB}(\theta)$ is a literal p or $\neg p$, we define $\chi_\varphi \coloneqq [E]\big(p_\varphi \to \varphi\big)$.

For the remaining subformulae φ of θ, with the exception of inclusion atoms, the formula χ_φ is defined as follows:

$$
\begin{aligned}
\chi_{\varphi\wedge\psi} &\coloneqq [E]\big((p_{\varphi\wedge\psi} \leftrightarrow p_\varphi) \wedge (p_{\varphi\wedge\psi} \leftrightarrow p_\psi)\big),\\
\chi_{\varphi\vee\psi} &\coloneqq [E]\big(p_{\varphi\vee\psi} \leftrightarrow (p_\varphi \vee p_\psi)\big).
\end{aligned}
$$

We then define the formulae χ_α where $\alpha \in \mathbf{SUB}(\theta)$ is an inclusion atom. We appoint a fresh binary relation R_α for each inclusion atom in θ. Assume α denotes the inclusion atom $p_1 \cdots p_k \subseteq q_1 \cdots q_k$. We define

$$
\begin{aligned}
\chi_\alpha^+ &\coloneqq \bigwedge_{i\,\in\,\{1,\ldots,k\}} [E]\big(\,(p_\alpha \wedge p_i) \;\to\; \langle R_\alpha\rangle(p_\alpha \wedge q_i)\,\big),\\
\chi_\alpha^- &\coloneqq \bigwedge_{i\,\in\,\{1,\ldots,k\}} [E]\big(\,(p_\alpha \wedge \neg p_i) \;\to\; \langle R_\alpha\rangle(p_\alpha \wedge \neg q_i)\,\big),\\
\chi_\alpha &\coloneqq \chi_\alpha^+ \wedge \chi_\alpha^- \wedge \bigwedge_{i\,\in\,\{1,\ldots,k\}} [E]\big(\,\langle R_\alpha\rangle q_i \to [R_\alpha] q_i\,\big).
\end{aligned}
$$

Finally, we define $\varphi_\theta \coloneqq p_\theta \wedge \bigwedge_{\varphi\,\in\,\mathbf{SUB}(\theta)} \chi_\varphi$.

Note that clearly the size of the formula φ_θ is polynomial with respect to the size of θ. The intuition why this translation works is as follows. We use auxiliary propositions p_φ for each formula $\varphi \in \mathbf{SUB}(\theta)$ so that their extensions correspond to teams where φ is true. The above clauses capture the truth conditions of team semantics so that, for example, $p_{\varphi\vee\psi}$ will ultimately correspond to a team where $\varphi \vee \psi$ is true iff both p_φ and p_ψ will correspond to teams where φ and ψ are true, respectively. The same is true for other operators in addition to $\vee$. Additionally, the translation has clauses that deal

with atomic formulae. The treatment on inclusion atoms in particular requires some intricate conditions to be satisfied. They have been dealt with via using the auxiliary binary predicates R_α, as will be established in the following proof.

Proof. We will show that any formula θ of propositional inclusion logic is satisfiable if and only if its translation φ_θ is. Furthermore, θ is satisfiable over a domain W if and only if φ_θ is satisfiable over W, and accordingly, we also get the desired result for finite satisfiability. $\mathcal{L}$ has the finite model property since it clearly translates to two-variable logic via a simple extension of the *standard translation* (see [BRV01] for the definition of standard translation).

Let $M = (W, R, V)$ be a Kripke model. Let $I(\theta) \subseteq \mathbf{SUB}(\theta)$ be the set of inclusion atoms in θ. Assume that $X \models \theta$, where X is a nonempty team. We next define a multimodal Kripke model $N := (W, R, \{R_\alpha\}_{\alpha \in I(\theta)}, V \cup U)$, where $U \colon \{ p_\varphi \mid \varphi \in \mathbf{SUB}(\theta)\} \to \mathcal{P}(W)$ extends the valuation function V. Define $U(p_\theta) = X$. Accordingly, we have $M, U(p_\theta) \models \theta$. Working from the root towards the leaves of the parse tree of θ, we next interpret the remaining predicates p_φ inductively such that the condition $M, U(p_\varphi) \models \varphi$ is maintained.

Assume $U(p_{\psi\wedge\psi'})$ has been defined. We define $U(p_\psi) = U(p_{\psi'}) = U(p_{\psi\wedge\psi'})$. As $U(p_{\psi\wedge\psi'}) \models \psi \wedge \psi'$, we have $U(p_\psi) \models \psi$ and $U(p_{\psi'}) \models \psi'$.

Assume then that $U(p_{\psi\vee\psi'})$ has been defined. As a result, there exist sets S and S' such that $S \models \psi$ and $S' \models \psi'$, and furthermore, $S \cup S' = U(p_{\psi\vee\psi'})$. We define $U(p_\psi) = S$ and $U(p_{\psi'}) = S'$.

We have now fixed an interpretation for each of the predicates p_φ. The relations R_α, where α is an inclusion atom, remain to be interpreted. Let $p_1 \cdots p_k \subseteq q_1 \cdots q_k$ be an inclusion atom in θ, and denote this atom by α. Call $T \coloneqq U(p_\alpha)$. Let $u \in T$. Since $T \models \alpha$, there exists a point $v \in T$ such that for each $i \in \{1, \ldots, k\}$, $u \in V(p_i)$ if and only if $v \in V(q_i)$. Define the pair (u, v) to be in R_α. In this fashion, consider each point u in T and find exactly one corresponding point v for u, and put the pair (u, v) into R_α. This fixes the interpretation of R_α.

Let $w \in X = U(p_\theta)$. Recalling how the sets $U(p_\varphi)$ were defined, it is now routine to check that $N, w \Vdash \varphi_\theta$.

We then consider the converse implication. Consequently, we assume that $N, w \Vdash \varphi_\theta$, where N is some multimodal Kripke model in the signature of φ_θ and w a point in the domain of N. We let W denote the domain and V the valuation function of N. For each $\varphi \in \mathbf{SUB}(\theta)$, define the team $X_\varphi \coloneqq V(p_\varphi)$. We will show by induction on the structure of θ that for each $\varphi \in \mathbf{SUB}(\theta)$, we have $X_\varphi \models \varphi$. Once this is done, it is clear that $X_\theta \models \theta$.

Furthermore, we have $X_\theta \neq \emptyset$ as $w \in V(p_\theta)$ (because $N, w \Vdash \varphi_\theta$). Now recall the definition of the formulae χ_φ, where $\varphi \in \mathbf{SUB}(\theta)$. Let $p \in \mathbf{SUB}(\theta)$. It is clear that $X_p \models p$, since $N, w \Vdash \chi_p$. Similarly, we infer that $X_{\neg q} \models \neg q$ for $\neg q \in \mathbf{SUB}(\theta)$.

Consider then a subformula $p_1 \cdots p_k \subseteq q_1 \cdots q_k$ of φ. Denote this inclusion atom by α. Consider a point $u \in X_\alpha$. If u satisfies p_i for some $i \in \{1, \ldots, k\}$, then we infer that since $N, w \Vdash \chi^+_\alpha$, there exists a point $v_i \in X_\alpha$ that satisfies q_i. Similarly, if u satisfies $\neg p_j$, we infer that since $N, w \Vdash \chi^-_\alpha$, there exists a point $v_j \in X_\alpha$ that satisfies $\neg q_j$. To conclude that $X_\alpha \models \alpha$, it suffices to show that all such points v_i and v_j can be chosen

such that $v_i = v_j$ for all $i, j \in \{1, \dots, k\}$. This follows due to the third conjunct of χ_α. Concerning the third conjunct, note here carefully that $[E]\big(\langle R_\alpha\rangle q_i \to [R_\alpha] q_i\big)$ implies that $[E]\big(\langle R_\alpha\rangle \neg q_i \to [R_\alpha]\neg q_i\big)$. Having established the basis of the induction, the rest of the argument is straightforward. □

Persistent subsets

Definition 40. Let $\mathfrak{A} = (A, S)$ be a structure with $A = \{1, \dots, n\}$ 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 $A = \{1, \dots, n\}$ and $S \subseteq A^3$
Question: exists some S-persistent set $P \subseteq A$ such that $n \in P$

Theorem 41 (closely related to 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 $A_C = \{1, \dots, 2^\ell\}$
- for all $i, j, k \in A$, let $(i, j, k) \in S_C$ if and only if C accepts the input tuple

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

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

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

Theorem 42 ([Hel+19, Lem. 3.4])**.** S-PER *is* EXP*-hard with respect to* $\leq_m^p$*-reductions.*

As a rough idea, one can show EXP-hardness by a generic reduction from alternating PSPACE Turing machines

EXP-hardness of PL[⊆]: prerequisites

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

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

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

Encoding S-PER **into** PL[⊆]
C is a Boolean circuit with 3ℓ input gates ordered $g_1, \dots, g_m$ such that $g_1, \dots, g_{3\ell}$ 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 PL[⊆] 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: $\psi_C \coloneqq \left(\bigwedge_{3\ell+1 \le i \le m} \theta_i\right) \wedge p_m$. (truth values of p_i match acc. computation of C)

Encoding persistency into the formula
Input gate propositions: $p_1, \dots, p_\ell, \underbrace{p_{\ell+1}, \dots, p_{2\ell}}_{=:q_1,\dots,q_\ell}, \underbrace{p_{2\ell+1}, \dots, p_{3\ell}}_{=:r_1,\dots,r_\ell}$ *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 S-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

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

"⇒" of the claim
In the following, we also present a proof in full details.

Proof. Let C be a Boolean circuit with 3ℓ input gates. Let $g_1, \dots, g_m$ be the gates of C, where $g_1, \dots, g_{3\ell}$ are the input gates and g_m is the output gate. We fix a distinct Boolean variable p_i for each gate g_i. Let Φ be the set $\{p_1, \dots, p_m\}$ of proposition symbols. We define for each $i \in \{3\ell+1, \dots, m\}$ a formula $\theta_i \in \mathrm{PL}[\subseteq](\Phi)$ that describes the correct operation of the gate g_i (where $\leftrightarrow$ is the usual shorthand for flat formulae):

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

Let ψ_C be the formula $\left(\bigwedge_{3\ell+1 \le i \le m} \theta_i\right) \wedge p_m$. That being so, ψ_C essentially says that the truth values of p_i, $1 \le i \le m$, match an accepting computation of C.

Now we can define a formula φ_C of PL[⊆](Φ) which is satisfiable if and only if C is a positive instance of S-PER. For the sake of readability, we denote here the variables

corresponding to the input gates $g_{\ell+1}, \dots, g_{2\ell}$ by $q_1, \dots, q_\ell$. Similarly, we denote the variables $p_{2\ell+1}, \dots, p_{3\ell}$ by $r_1, \dots, r_\ell$.

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

Note that φ_C can clearly be constructed from the circuit C in polynomial time.

Assume first that φ_C is satisfiable. That being so there is a nonempty team T such that $T \models \varphi_C$. Consider the model $\mathfrak{A}_C = (A_C, S_C)$ that corresponds to the circuit C. We define a subset P of A_C as follows: $P \coloneqq \{\mathrm{bin}(a_1 \dots a_\ell) \mid (a_1 \dots a_\ell) \in T(p_1, \dots, p_\ell)\}$.

Observe first that since $T \models p_m$ and $T \models p_m \cdots p_m \subseteq p_1 \cdots p_\ell$, $(1, \dots, 1) \in T(p_1, \dots, p_\ell)$ and that being so $2^\ell = \mathrm{bin}(1 \dots 1) \in P$. On that account, it suffices to show that P is S_C-persistent. To prove this, assume that $i = \mathrm{bin}(a_1 \dots a_\ell) \in P$. Then there is a state $w \in T$ such that $w \in V(p_t) \iff a_t = 1 \quad \text{for } 1 \leq t \leq \ell$.

Define now $b_t, c_t \in \{0, 1\}$, $1 \leq t \leq \ell$, by the condition

$$b_t = 1 \iff s(q_t) = 1 \quad \text{and} \quad c_t = 1 \iff s(r_t) = 1.$$

As $T \models \psi_C$, it follows from flatness that $s \models \psi_C$ for every $s \in T$. By the definition of ψ_C, this means that the circuit C accepts the input tuple $(a_1, \dots, a_\ell, b_1, \dots, b_\ell, c_1, \dots, c_\ell)$. That being the case, $(i, j, k) \in S_C$, where $j = \mathrm{bin}(b_1 \dots b_\ell)$ and $k = \mathrm{bin}(c_1 \dots c_\ell)$.

We still need to show that $j, k \in P$. To see this, note that since $T \models q_1 \cdots q_\ell \subseteq p_1 \cdots p_\ell$, there exists $s \in T$ such that

$$s(p_t) = 1 \iff s(q_t) = 1 \iff b_t = 1 \quad \text{for } 1 \leq t \leq \ell.$$

Accordingly, $(b_1, \dots, b_\ell) \in T(p_1, \dots, p_n)$, and on that account $j \in P$. Similarly we see that $k \in P$.

To prove the other implication, assume that C is a positive instance of the problem S-PER. Then there is an S_C-persistent set $P \subseteq A_C$ such that $2^\ell \in P$. We let T the team such that T is the set of all tuples $(a_1, \dots, a_m) \in \{0, 1\}^m$ that correspond to an accepting computation of C and for which $\mathrm{bin}(a_1 \dots a_\ell)$, $\mathrm{bin}(a_{\ell+1} \dots a_{2\ell})$, $\mathrm{bin}(a_{2\ell+1} \dots a_{3\ell}) \in P$.

We will now show that $T \models \varphi_C$, and accordingly φ_C is satisfiable. Note first that $T \models \psi_C$, since by the definition of T, for any $s \in T$, the truth values of p_i w.r.t. s correspond to an accepting computation of C.

To prove $T \models q_1 \cdots q_\ell \subseteq p_1 \cdots p_\ell$, assume that $(b_1, \dots, b_\ell) \in T(q_1, \dots, q_\ell)$. Then $i \coloneqq \mathrm{bin}(b_1 \dots b_\ell) \in P$, and since P is S_C-persistent, there are $j, k \in P$ such that $(i, j, k) \in S_C$. Accordingly, there is a tuple $(a_1, \dots, a_m) \in \{0, 1\}^m$ corresponding to an accepting computation of C such that $(a_1, \dots, a_\ell) = (b_1, \dots, b_\ell)$, $j = \mathrm{bin}(a_{\ell+1} \dots a_{2\ell})$ and $k = \mathrm{bin}(a_{2\ell+1} \dots a_{3\ell})$. This means that $(a_1, \dots, a_m)$ is in T, and that being the case $(b_1, \dots, b_\ell) \in T(p_1, \dots, p_\ell)$. The claim $M, T \models r_1 \cdots r_\ell \subseteq p_1 \cdots p_\ell$ is proved in the same way.

Note that since $T \models p_m$, we have $T(p_m, \dots, p_m) = \{(1, \dots, 1)\}$. Furthermore, since $2^\ell = \mathrm{bin}(1 \dots 1) \in P$ and P is S_C-persistent, there is an element $(a_1, \dots, a_m) \in T$ such that $(a_1, \dots, a_\ell) = (1, \dots, 1)$. Consequently, we see that $(1, \dots, 1) \in T(p_1, \dots, p_\ell)$, and as a result $T \models p_m \cdots p_m \subseteq p_1 \cdots p_\ell$. □

Conclusion of Lecture 3

- PL[⊆]-MC is P-complete.
- PL[⊆]-SAT is EXP-complete.
- PL[⊆]-VAL is coNP-complete.

4 Complexity of propositional dependence logic and beyond

Literature: [Vir17; Han+18]

In this section, we explore the complexity of validity and model checking for PL[dep].

Canonical complete problems: SAT and QBF

SAT [Coo71]		QBF (Stockmeyer and Meyer, 1973)	
Input:	Boolean formula θ	Input:	Quantified Boolean formula $\phi \coloneqq Q_1 p_1 \dots Q_n p_n \theta$
Question:	Is θ satisfiable?	Question:	Is ϕ true?
Complete for:	NP (Thm. 5)	Complete for:	PSPACE

W.l.o.g. θ in 3CNF

$$\theta = (p_1 \vee p_2 \vee \neg p_3) \wedge (\neg p_2 \vee \neg p_4 \vee p_5) \wedge \dots$$

Model Checking for Dependence Logic

Theorem 43 ([Loh12, Theorem 4.13])**.** PL[dep]-MC *is* NP*-complete.*

Proof ideas:

Membership: Use nondeterminism for splitjunctions.

Hardness: reduce from 3SAT.

Membership in NP

Proof. The following recursive algorithm CHECK(formula ϕ, team T) shows membership in NP.

1. switch ϕ:
2. case $\phi = p \in \mathsf{PROP}$: foreach $s \in T$ do if not $s(p)$ then return 0 endfor return 1.
3. case $\phi = \neg p, p \in \mathsf{PROP}$: foreach $s \in T$ do if $s(p)$ then return 0 endfor return 1.
4. case $\phi = \mathrm{dep}(P, Q)$: foreach $(s, t) \in T \times T$ do if $s{\restriction}P = t{\restriction}P$ and $s{\restriction}Q \neq t{\restriction}Q$ then return 0 endfor return 1.

5. case $\phi = \psi \wedge \theta$: return $\textsc{Check}(\psi, T) \wedge \textsc{Check}(\theta, T)$.
6. case $\phi = \psi \vee \theta$: guess T_1, T_2 with $T_1 \cup T_2 = T$, if $T_1 \cup T_2 \neq T$ then return 0 else return $\textsc{Check}(\psi, T_1) \wedge \textsc{Check}(\theta, T_2)$.

Runtime of one call is $O(||T|| + ||\phi|| + ||T||^2)$, where $||\cdot||$ denotes the encoding length. We have $|\phi|$ recursion depth. Together still polynomial in the input length. Correctness via induction on formula structure. □

NP Lower Bound

Proof. Show 3SAT $\leq_m^p$ PL[dep]-MC. Let $\phi = \bigwedge_{i=1}^m (\ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3})$ be a 3CNF formula over variables $\{x_1, \ldots, x_n\}$. Assume without loss of generality, that ϕ does not contain any tautological clauses (i.e., a clause where a variable x and its negation $\neg x$ is contained). We define the team $T = \{s_1, \ldots, s_m\}$ over variables $\{v_1, \ldots, v_n, p_1, \ldots, p_n\}$, where the intuitive meaning is that every assignment encodes a clause. The assignment s_i maps v_j to 1, if variable x_j occurs in clause i with parity $s_i(p_j)$. Formally

$$s_i(v_j) = \begin{cases} 1, & \text{if } x_j \text{ is in clause } i \\ 0, & \text{otherwise.} \end{cases}$$

$$s_i(p_j) = \begin{cases} 1, & \text{if } x_j \text{ is positive in clause } i \\ 0, & \text{otherwise.} \end{cases}$$

Note that the otherwise case for $s_i(p_j)$ contains the case where neither x_j nor $\neg x_j$ are in clause i. Let $\psi \coloneqq \bigvee_{j=1}^n (v_j \wedge \mathrm{dep}(\{p_j\}))$. We claim that ϕ is satisfiable iff $T \models \psi$.

"⇒": Let I be a satisfying assignment for ϕ. Now note that since $I(\phi) = 1$, each clause is made true by some variable x_j. So we split the assignments (that correspond to clauses) into groups $T_1, \ldots, T_n$ such that in T_i are those clauses that x_i makes true. Formally construct $T_1, \ldots, T_n \subseteq T$:

$$T_j \coloneqq \{s_i \in T \mid s_i(v_j) = 1, s_i(p_j) = I(x_j)\}$$

Since $I(\phi) = 1$, each clause is made true by some variable x_j with the parity $I(x_j)$ and thus $T_1 \cup \cdots \cup T_n = T$. Now, by the selection of T_j, $T_j \models v_j \wedge \mathrm{dep}(\{p_j\})$ for $1 \leq j \leq n$ and hence $T \models \psi$.

"⇐": Now, assume $T \models \psi$. Hence, $T = T_1 \cup \cdots \cup T_n$ with $T_j \models v_j \wedge \mathrm{dep}(\{p_j\})$ for $1 \leq j \leq n$. We will show how to construct a satisfying assignment I for ϕ from $T_1, \ldots, T_n$. Let b_j denote the constant value given for p_j in T_j. If $T_j = \emptyset$, we set $b_j \coloneqq 1$. We then define I such that

$$I(x_j) = b_j.$$

We have to prove that I is well defined, and that $I(\phi) = 1$. It is immediate that I is well defined as the definition sets the truth value of every variable uniquely once. It suffices to show that I makes every clause of ϕ true. Let C_i be some clause of ϕ and s_i its corresponding assignment. We know that $s_i \in T_j$, for some j. Since $T_j \models v_j \wedge \mathrm{dep}(\{p_j\})$, we obtain that x_j occurs in C_i with parity b_j. Thus I makes C_i true.

□

Canonical complete problems: DQBF

DQBF (Peterson, Reif, Azhar, 2001)	
Input:	Dependency Quantified Boolean formula $\phi \coloneqq \forall p_1 \dots \forall p_m \exists q_1 \dots \exists q_n \theta$ and constraints $\vec{c}_1, \dots, \vec{c}_n$
Question:	Is ϕ true?
Complete for:	NEXPTIME

- The constraint $\vec{c}_i$ is a tuple of the universally quantified variables of which the existentially quantified variable q_i may depend on.
- A DQBF formula $\forall p_1 \dots \forall p_m \exists q_1 \dots \exists q_n \theta$ with constraints $\vec{c}_1, \dots, \vec{c}_n$ is true, if the the following formula with Boolean function quantification

$$\exists f_1 \dots f_n \forall p_1 \dots \forall p_m \theta(f_1(\vec{c}_1)/q_1, \dots f_n(\vec{c}_n)/q_n)$$

  is true. Note that f_i is a Boolean function (Skolem function) which is used to interpret q_i given the values of the variables in $\vec{c}_i$.
- Note how close the above is to $\mathrm{dep}(\vec{c}_1, q_1) \wedge \dots \wedge \mathrm{dep}(\vec{c}_n, q_n) \wedge \theta$!

As hinted above DQBF and PL[dep] are closely connected. Next we will show that the validity problem for PL[dep] is NEXPTIME complete by a reduction from DQBF, which uses dependence atoms to describe constraints.

The validity problem for PD is in NEXPTIME

We start with a simple lemma that reduces validity of PL[dep] to model checking on an exponentially large team.
If $D \subseteq \mathsf{PROP}$, we denote by 2^D the set of all assignments $s\colon D \to \{0,1\}$.

Lemma 44. *A* PL[dep]*-formula φ with proposition symbols in D is valid iff $2^D \models \varphi$.*

Proof. Left-to-right direction is trivial and the converse follows from downward closure. □

Lemma 45. *The validity problem for* PL[dep] *is in NEXPTIME.*

Proof. Let $\varphi \in$ PL[dep] whose variables are in D. By Lemma 44, φ is valid iff $2^D \models \varphi$. The size of 2^D is $2^{|D|} \leq 2^{|\varphi|}$. Therefore 2^D can be constructed from φ in exponential time. By Theorem 43, there exists an NP algorithm (with respect to $|2^D| + |\varphi|$) for checking whether $2^D \models \varphi$. Clearly this algorithm is in NEXPTIME with respect to $|\varphi|$. □

The validity problem for PD is NEXPTIME-hard

We will associate each DQBF-formula μ with a corresponding PL[dep]-formula φ_μ. Let

$$\mu = \big(\forall p_1 \dots \forall p_n \exists q_1 \dots \exists q_k \, \theta, (\vec{c}_1, \dots, \vec{c}_k)\big)$$

be a DQBF-formula and denote by D_μ the set of propositional variables in μ, i.e., $D_\mu \coloneqq \{p_1, \dots, p_n, q_1, \dots, q_k\}$. For each tuple of propositional variables $\vec{c}_i$, $i \leq k$, we stipulate that $\vec{c}_i = (p_{i_1}, \dots, p_{i_{n_i}})$. Thus n_i denotes the lenth of $\vec{c}_i$. Define

$$\varphi_\mu \coloneqq \theta \vee \bigvee_{i \leq k} \mathrm{dep}(p_{i_1}, \dots, p_{i_{n_i}}, q_i).$$

We will show that μ is true if and only if the PL[dep]-formula φ_μ is valid. By Lemma 44, it suffices to show that μ is valid if and only if $2^{D_\mu} \models \varphi_\mu$. Since DQBF is NEXPTIME-complete and φ_μ is polynomial with respect to μ, it follows that the validity problem for PL[dep] is NEXPTIME-hard.

Proof of correctness. Assume first that μ is valid, i.e., that $\forall p_1 \dots \forall p_n \exists q_1 \dots \exists q_k\, \theta$ is valid under the constraint $(\vec{c}_1, \dots, \vec{c}_k)$. Therefore, for each $i \leq k$, there exists a function $f_i : \{0,1\}^{n_i} \to \{0,1\}$ such that

$$\text{for every assignment } s : \{p_1, \dots, p_n\} \to \{0,1\} : \quad s' \models \theta, \tag{4.1}$$

where $s' \coloneqq s\Big(q_1 \mapsto f_1\big(s(\vec{c}_1)\big), \dots, q_k \mapsto f_k\big(s(\vec{c}_k)\big)\Big)$. Our goal is to show that

$$2^{D_\mu} \models \theta \vee \bigvee_{i \leq k} \mathrm{dep}(p_{i_1}, \dots, p_{i_{n_i}}, q_i).$$

It suffices to show that there exist some $Y, Z_1, \dots Z_k \subseteq 2^{D_\mu}$ such that $Y \cup Z_1 \cup \dots \cup Z_k = 2^{D_\mu}$, $Y \models \theta$, and $Z_i \models \mathrm{dep}(p_{i_1}, \dots, p_{i_{n_i}}, q_i)$, for each $i \leq k$. We define the team Z_i, for each $i \leq k$, by using the function f_i. Define

$$Z_i \coloneqq \{s \in 2^{D_\mu} \mid s(q_i) \neq f_i\big(s(p_{i_1}), \dots, s(p_{i_{n_i}})\big)\}, \quad \text{for each } i \leq k.$$

Since propositional variables have only 2 possible values, we conclude that, for each $i \leq k$, $Z_i \models \mathrm{dep}(p_{i_1}, \dots, p_{i_{n_i}}, q_i)$. Thus

$$\bigcup_{1 \leq i \leq k} Z_i \models \bigvee_{i \leq k} \mathrm{dep}(p_{i_1}, \dots, p_{i_{n_i}}, q_i). \tag{4.2}$$

Note that $s(q_i) = f_i\big(s(p_{i_1}), \dots, s(p_{i_{n_i}})\big)$ holds for every $s \in (2^{D_\mu} \setminus Z_i)$ and every $i \leq k$. Define then

$$Y \coloneqq 2^{D_\mu} \setminus \bigcup_{1 \leq i \leq k} Z_i.$$

Clearly, for every $s \in Y$ and $i \leq k$, it holds that $s(q_i) = f_i\big(s(p_{i_1}), \dots, s(p_{i_{n_i}})\big)$. Recall that, for each $i \leq k$, $\vec{c}_i = (p_{i_1}, \dots, p_{i_{n_i}})$. Thus from (4.1), it follows that $s \models \theta$, for every $s \in Y$. Since θ is a PL-formula, we conclude by flatness of PL that $Y \models \theta$. From this together with (4.2), we conclude that $2^{D_\mu} \models \varphi_\mu$.

Assume then that $2^{D_\mu} \models \varphi_\mu$. Thus there exist some $Y_1, \dots, Y_k, Z$ such that $Y_1 \cup \dots \cup Y_k \cup Z = 2^{D_\mu}$, $Z \models \theta$, and

$$Y_i \models \mathrm{dep}(p_{i_1}, \dots, p_{i_{n_i}}, q_i), \text{ for each } i \leq k.$$

Assume that we have picked $Y_1, \ldots, Y_k, Z$ such that Z is minimal. We will show that then $Z \models \mathrm{dep}(p_{i_1}, \ldots, p_{i_{n_i}}, q_i)$, for each $i \leq k$. Assume for the sake of a contradiction that, for some $i \leq k$, there exist $s, t \in Z$ such that

$$s(p_{i_1}) = t(p_{i_1}), \ldots, s(p_{i_{n_i}}) = t(p_{i_{n_i}}) \text{ but } s(q_i) \neq t(q_i).$$

Clearly either $Y_i \cup \{s\} \models \mathrm{dep}(p_{i_1}, \ldots, p_{i_{n_i}}, q_i)$ or $Y_i \cup \{t\} \models \mathrm{dep}(p_{i_1}, \ldots, p_{i_{n_i}}, q_i)$. This contradicts the fact that Z was assumed to be minimal.

We will then show that, for every $a_1, \ldots, a_n \in \{0,1\}$, there exists some assignment s in Z that expands

$$(p_1, \ldots, p_n) \mapsto (a_1, \ldots, a_n).$$

Fix $a_1, \ldots, a_n \in \{0,1\}$. Now, for every $i \leq k$, since $Y_i \models \mathrm{dep}(p_{i_1}, \ldots, p_{i_{n_i}}, q_i)$, it follows that for any two $s', s'' \in Y_i$ that expand $(p_1, \ldots, p_n) \mapsto (a_1, \ldots, a_n)$, it holds that $s'(q_i) = s''(q_i)$. Thus, for each $i \leq k$, there exists a truth value $b_i \in \{0,1\}$ such that there is no expansions of $(p_1, \ldots, p_n, q_i) \mapsto (a_1, \ldots, a_n, b_i)$ in Y_i. Therefore, the assignment $(p_1, \ldots, p_n, q_1, \ldots, q_k) \mapsto (a_1, \ldots, a_n, b_1, \ldots, b_k)$ is not in Y_i, for any $i \leq k$. Thus the assignment $(p_1, \ldots, p_n, q_1, \ldots, q_k) \mapsto (a_1, \ldots, a_n, b_1, \ldots, b_k)$ is in Z. Hence, for every $a_1, \ldots, a_n \in \{0,1\}$, there exists some expansion of $(p_1, \ldots, p_n) \mapsto (a_1, \ldots, a_n)$ in Z.

Now, for each $i \leq k$, we define a function $f_i : \{0,1\}^{n_i} \to \{0,1\}$ as follows. Define

$$f_i(a_1, \ldots, a_{n_i}) \coloneqq s(q_i),$$

where s is an assignment in Z that expands $(p_{i_1}, \ldots p_{i_{n_i}}) \mapsto (a_1, \ldots, a_{n_i})$. Since $Z \models \mathrm{dep}(p_{i_1}, \ldots, p_{i_{n_i}}, q_i)$, for each $i \leq k$, the functions f_i are well defined. Recall that, for each $i \leq k$, $\vec{c}_i = (p_{i_1}, \ldots p_{i_{n_i}})$. Now since θ is syntactically a PL-formula and since $Z \models \theta$, it follows from flatness of PL that $s' \models \theta$, for each $s' \in Z$. Clearly the functions f_i, for $i \leq k$, are as required in (4.1). Thus we conclude that (4.1) holds. Thus μ is valid. □

A logic to rule them all

The extension of PL with the contradictory negation PL[∼]

$$X \models {\sim}\varphi \iff X \not\models \varphi$$

is very expressive and all connectives studied in team sematics can be defined in it.

The connectives below can be defined in PL[∼] with polynomial blow up.

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

Also dependence/inclusion/independence atoms can be expressed in PL[∼] with polynomial blow up [LV19].

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

Interestingly, the satisfiability and validity problems for PTL are interdefinable with polynomial size definitions. In particular, this implies that there is not much difference between SAT and VAL in PTL-setting.

PTIME **Reductions Between Validity and Satisfiability**
Note: $X \models \sim(p \wedge \neg p)$ iff X is non-empty.

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

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

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

Theorem 46.
- *φ is satisfiable iff φ_{SAT} is valid.*
- *φ is valid iff φ_{VAL} is satisfiable.*

Oracle Turing Machines
The exponential-time hierarchy corresponds to the class of problems that can be recognized by an exponential-time alternating Turing machine with constantly many alternations.

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

Orponen's characterization can be generalised to exponential-time alternating Turing machines with polynomially many alternations (i.e. the class $\mathsf{AEXPTIME}(\mathsf{poly})$) by allowing queries to polynomially many oracles.

Idea: SAT for PL[$\sim$] **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 PL[$\sim$].

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 47. *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,*

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

where n is the length of x and $Q_1, \dots, Q_{f(n)}$ 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 48. *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*

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

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 SAT(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:

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

In 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 the statement

$$\exists s \in X \text{ s.t. } \{s\} \models \vec{q} = \vec{a} \wedge \vec{r} = \mathrm{bin}(i).$$

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

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

In PL[$\sim$] this is written as

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

Simulating Quantification
Recall:

- The whole computation is encoded in a team.
- The decoding uses statements: $\exists s \in X$ 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 to be a huge team consisting of all assignments for variables.
- $\exists Y\varphi(Y) \quad \mapsto \quad \exists A \subseteq D$: $(D \setminus A) \models \varphi$.
- $\forall Y\varphi(Y) \quad \mapsto \quad \forall A \subseteq D$: $(D \setminus A) \models \varphi$.
- Maintain uniformity in quantification. (Arities of A and D do not coincide.)

Example of Quantification

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

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 $\bot_c$ needed)

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

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

Complexity of PL[$\sim$]

Theorem 49. SAT(PL[$\sim$]) *is* AEXPTIME(poly)*-complete.*

Proof. Hardness: By simulating polynomial time alternating oracle Turing machines. Membership: Guess a possibly exponential-size team T and do APTIME model checking. □

Corollary 50. VAL(PL[$\sim$]) *is* AEXPTIME(poly)*-complete.*

Theorem 51. MC(PL[$\sim$]) *is* PSPACE*-complete*

Complexity Results

Logic	SAT	VAL	MC
PL	NP [0]	coNP [0]	NC[1] [1]
PL[dep]	NP [3]	NEXPTIME [4]	NP [2]
PL[$\bot_c$]	NP [7]	in coNEXPTIME$^{\mathsf{NP}}$ [7]	NP [7]
PL[$\subseteq$]	EXP [5]	coNP [7]	in P [6]
PL[$\sim$]	AEXPTIME(poly) [7]	AEXPTIME(poly) [7]	PSPACE [8]

[0] Cook 1971, Levin 1973, [1] Buss 1987, [2] Ebbing, Lohmann 2012, [3] Lohmann, Vollmer 2013, [4] Virtema 2014, [5] Hella, Kuusisto, Meier, Vollmer 2015, [6] Hella, Kuusisto, Meier and Virtema 2019, [7] Hannula, Kontinen, Virtema, Vollmer 2018, [8] Müller 2014.

Conclusion of Lecture 4

- DQBF is a canonical NEXPTIME-complete problem.
- SAT(PL[dep]) and MC(PL[dep]) are NP-complete.

- VAL(PL[dep]) is NEXPTIME-complete.
- SAT(PL[$\sim$]) and VAL(PL[$\sim$]) are AEXPTIME(poly)-complete.
- MC(PL[$\sim$]) are PSPACE-complete.

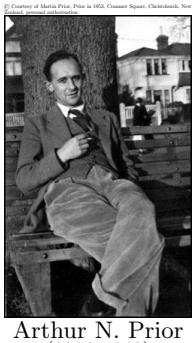
5 Recent Trends: Hyperproperties

Literature: [Vir+21; Gut+22]

In this lecture, we give an introduction into a recent trend in the field of team semantics, namely the study of hyperproperties. First, we will explain syntax and semantics of classical temporal logics (LTL, CTL, and CTL*). Afterwards, we give a thorough motivation of hyperproperties and their importance in the context of verification and specification of concurrent systems. We will show how these properties can be expressed in a logic with team semantics.

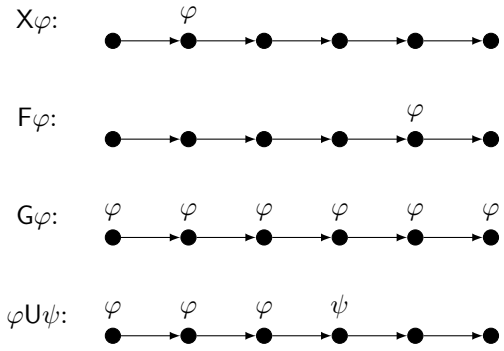
Temporal Logic

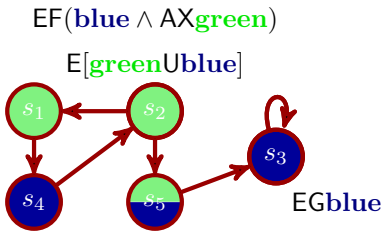
- Dates back to Arthur Norman Prior (1914–1969)
- New modalities neXt, Until, Future, Global
- and quantifiers: All paths, Exists a path
- 'From Philosophical to Industrial Logics' [Var09]



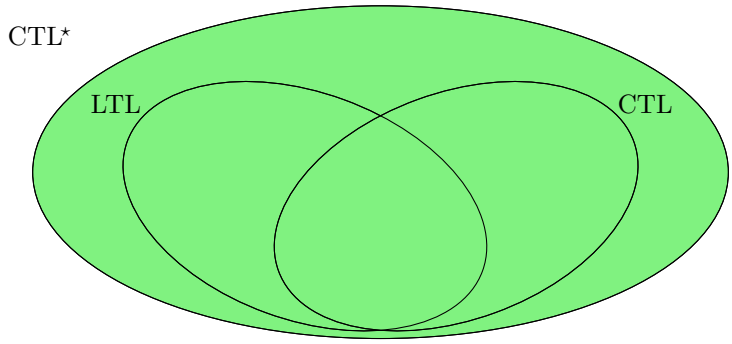
Arthur N. Prior
(1914–1969)

Temporal Logic: Semantics by Example





A Temporal Lanscape of Logics



CTL*: Syntax

$$\varphi ::= \top \mid x \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathsf{X}\varphi \mid \varphi\mathsf{U}\varphi \mid \mathsf{E}\varphi,$$

LTL: Formulae $\mathsf{E}\varphi$ with

$$\varphi ::= \top \mid x \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathsf{X}\varphi \mid \varphi\mathsf{U}\varphi,$$

CTL: Syntax

$$\varphi ::= \top \mid x \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathsf{EX}\varphi \mid \mathsf{E}[\varphi\mathsf{U}\varphi] \mid \mathsf{AF}\varphi,$$

where $x \in \mathsf{PROP}$.

State and Path Formulae

Regarding the formulae of CTL*, we follow the terminology of Emerson and Sistla [ES84].

S1: Any atomic proposition is a state formula.

S2: If ψ, φ are state formula, then so are $\varphi \wedge \psi$, and $\neg\psi$.

S3: If ψ is a path formula, then $\mathsf{E}\psi$ is a state formula.

P1: Any state formula is a path formula.

P2: If ψ, φ are path formulae, then so are $\varphi \wedge \psi$, $\neg\psi$.

P3: If ψ, φ are path formulae, then so are $\mathsf{X}\varphi$, $[\varphi\mathsf{U}\psi]$.

Intuitively, (S1), (P1), (P2), and (P3) form LTL.

Kripke Frames and Paths

Definition 52. A *frame* $\mathcal{F}$ is a tuple $\mathcal{F} = (W, R)$, where W is a set of *worlds* and R is its *transition relation*, i.e., $R \subseteq W \times W$ and R is total (every state has ≤ 1 successor).

Definition 53. Let PROP be an countably infinite set of symbols. A *model* is a tuple $\mathcal{M} = (\mathcal{F}, V)$, where $\mathcal{F} = (W, \mathbf{R})$ is a frame and $V \colon \mathsf{PROP} \to \mathfrak{P}(W)$ is a *labeling* function.

Definition 54. A *path* $\pi = w_0, w_1, \ldots$ in a frame (W, R) is an infinite sequence of states such that $(w_i, w_{i+1}) \in R$ for all i. For $\pi = w_0, w_1, \ldots$, let $\pi_i \coloneqq w_i w_{i+1} \ldots$, $\pi[i] \coloneqq w_i$.

In the following, we define semantics for CTL* which encompasses the semantics for the syntactic restrictions CTL and LTL. Here the first four rows define semantics for state formulae while the rest define semantics for path formulae.

Formal Semantics of Temporal Logics

Definition 55. Let $\mathcal{M} = (W, R, V)$ be a model, $w \in W$, π be a path in (W, R).

$$
\begin{aligned}
\mathcal{M}, w &\models p \text{ iff } w \in V(p),\\
\mathcal{M}, w &\models \neg\varphi \text{ iff } \mathcal{M}, w \not\models \varphi,\\
\mathcal{M}, w &\models \varphi \wedge \psi \text{ iff } \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi,\\
\mathcal{M}, w &\models \mathsf{E}\varphi \text{ iff there is a path } \pi \text{ with } \pi[0] = w \text{ we have that } \mathcal{M}, \pi \models \varphi,\\
\mathcal{M}, \pi &\models p \text{ iff } \pi[0] \in V(p),\\
\mathcal{M}, \pi &\models \neg\varphi \text{ iff } \mathcal{M}, \pi \not\models \varphi,\\
\mathcal{M}, \pi &\models \varphi \wedge \psi \text{ iff } \mathcal{M}, \pi \models \varphi \text{ and } \mathcal{M}, \pi \models \psi,\\
\mathcal{M}, \pi &\models \mathsf{X}\varphi \text{ iff } \mathcal{M}, \pi_1 \models \varphi,\\
\mathcal{M}, \pi &\models \varphi \mathsf{U} \psi \text{ iff there is a } k \geq 0 \text{ s.t. } \mathcal{M}, \pi[k] \models \psi \text{ and}\\
&\qquad\qquad \text{for all } i \leq k \text{ we have that } \mathcal{M}, \pi[i] \models \varphi.
\end{aligned}
$$

Remaning Operators by Short Cuts

There exist further operators which can be defined by the operators we have already defined:

$$
\begin{aligned}
\mathsf{A}\varphi &\coloneqq \neg\mathsf{E}\neg\varphi\\
\mathsf{F}\varphi &\coloneqq [\top \mathsf{U} \varphi]\\
\mathsf{G}\varphi &\coloneqq \neg\mathsf{F}\neg\varphi\\
[\varphi \mathsf{W} \psi] &\coloneqq \neg[\neg\varphi \mathsf{U} \neg\psi]
\end{aligned}
$$

Two Important Problems and their Complexities

Satisfiability:

Given: a formula φ

Question: Is φ satisfiable?

Model Checking:

Given: a formula φ and a model $\mathcal{M}$

Question: Is there a $w \in \mathcal{M}$ that satisfies φ?

	SAT	MC
CTL*	EEXP[1]	PSPACE[2]
LTL	PSPACE[3]	PSPACE[4]
CTL	EXP[5]	P[6]

Examples for Interesting Properties in Temporal Logics

Mutal exclusion, i.e., no two processes can be in their critical section at the same time:

$$\mathsf{AG}(\neg p_1 \vee \neg p_2)$$

Starvation freeness, i.e., there is always a call to process p:

$$\mathsf{AF}p$$

Progress, i.e., some property r which implies a future call of process p:

$$\mathsf{AG}(r \rightarrow \mathsf{AF}p)$$

Logics for verification and specification of concurrent systems

Basic setting:

- System (e.g., piece of software or hardware) $\rightsquigarrow$ Kripke structure depicting the behaviour of the system
- A single run of the system $\rightsquigarrow$ a trace generated by the Kripke structure
- A property of the system (e.g., every request is eventually granted) $\rightsquigarrow$ a formula of some formal language expressing the property.

Model checking:

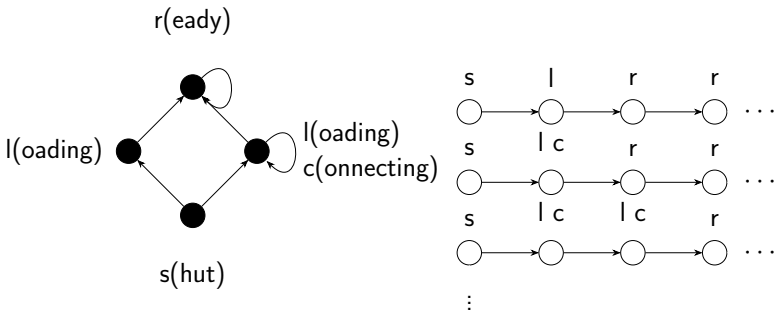
- Check whether a given system satisfies a given specification.

SAT solving:

- Check whether a given specification (or collection of) can be realised.

Traceproperties and hyperproperties

Opening your office computer after holidays:



Traceproperties hold in a system if each trace (in isolation) has the property:

- The computer will be eventually ready (or will be loading forever).

Hyperproperties are properties of sets of traces:

- The computer will be ready in bounded time.

Logics for traceproperties

- Linear-time temporal logic (LTL) is one of the most prominent logics for the specification and verification of reactive and concurrent systems.
- Model checking tools like SPIN and NuSMV automatically verify whether a given computer system is correct with respect to its LTL specification.
- One reason for the success of LTL over first-order logic is that LTL is a purely modal logic and thus has many desirable properties.
  - LTL is decidable (PSPACE-complete model checking and satisfiability).
  - $\mathrm{FO}^2(\leq)$ and $\mathrm{FO}^3(\leq)$ SAT are NEXPTIME-complete and non-elementary.
- Caveat: LTL can specify only traceproperties.

Recipe for logics for hyperproperties

A logic for traceproperties $\rightsquigarrow$ add trace quantifiers

In LTL the satisfying object is a trace: $T \models \varphi$ iff $\forall t \in T : t \models \varphi$

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \vee \varphi) \mid X\varphi \mid \varphi U \varphi$$

In HyperLTL the satisfying object is a set of traces and a trace assignment: $\Pi \models_T \varphi$

$$\varphi ::= \exists \pi \varphi \mid \forall \pi \varphi \mid \psi$$
$$\psi ::= p_\pi \mid \neg\psi \mid (\psi \vee \psi) \mid X\psi \mid \psi U \psi$$

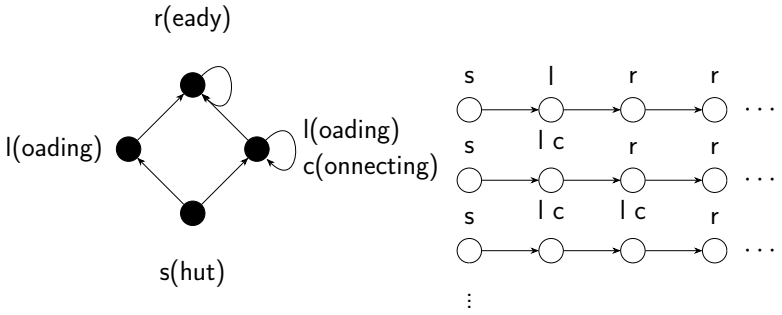
HyperQPTL extends HyperLTL by (uniform) quantification of propositions: $\exists p\varphi$, $\forall p\varphi$

Hyperlogics via quantifier extensions

- LTL, QPTL, CTL, etc. vs. HyperLTL, HyperQPTL, HyperCTL, etc. are prominent logics for traceproperties vs. hyperproperties of systems
  - Traceproperty: Each request is eventually granted (properties of traces)
  - Hyperproperty: Non-inference (values of public outputs do not leak information about confidential bits), (properties of sets of traces)
- HyperLogics are of high complexity or undecidable. Not well suited for properties involving unbounded number of traces.

Properties of quantification based hyperproperties

- Quantification based logics for hyperproperties: HyperLTL, HyperCTL, etc.
- Retain some desirable properties of LTL, but are not purely modal logics
  - Model checking for $\exists^*$HyperLTL and HyperLTL are PSPACE and non-elementary.
  - HyperLTL satisfiability is highly undecidable.
  - HyperLTL formulae express properties expressible using fixed finite number of traces.
- Bounded termination is not definable in HyperLTL (but is in HyperQPTL)



- Team semantics is a candidate for a purely modal logic without the above caveat.

Hyperlogic via team semantics

- Temporal logics with team semantics express hyperproperties.
- Purely modal logic & well suited for properties of unbounded number of traces.
- Expressivity
  - TeamLTL and HyperLogics are othogonal in expressivity.
  - Well behaved fragments of TeamLTL can be translated to HyperLogics with some form of set quantification.
  - Upper bound of expressivity is often monadic second-order logic with equi-level predicate.
- Complexity:
  - Where is the undecidability frontier of TeamLTL extensions?
  - A large EXPTIME fragment: left-flat and downward closed logics
  - Already TeamLTL with inclusion atoms and Boolean disjunctions is undecidable

LTL, HyperLTL, and TeamLTL

In LTL the satisfying object is a trace: $T \models \varphi$ iff $\forall t \in T : t \models \varphi$

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \vee \varphi) \mid X\varphi \mid \varphi U \varphi$$

In HyperLTL the satisfying object is a set of traces and a trace assignment: $\Pi \models_T \varphi$

$$\varphi ::= \exists \pi \varphi \mid \forall \pi \varphi \mid \psi$$
$$\psi ::= p_\pi \mid \neg\psi \mid (\psi \vee \psi) \mid X\psi \mid \psi U \psi$$

In TeamLTL the satisfying object is a set of traces. We use team semantics: $(T, i) \models \varphi$

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

+ new atomic statements (dependence and inclusion atoms: $\mathrm{dep}(\vec{p}, q)$, $\vec{p} \subseteq \vec{q}$) + additional connectives (Boolean disjunction, contradictory negation, etc.)

Extensions are a well-defined way to delineate expressivity and complexity

Semantics of TeamLTL

Temporal team semantics is universal and synchronous

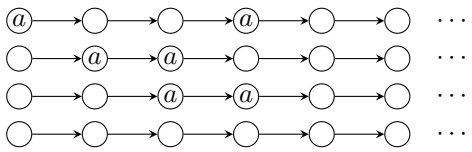
$$(T, i) \models p \text{ iff } \forall t \in T : t[i](p) = 1 \qquad (T, i) \models \neg p \text{ iff } \forall t \in T : t[i](p) = 0$$

$$(T, i) \models \mathsf{F}\varphi \text{ iff } (T, j) \models \varphi \text{ for some } j \geq i \qquad (T, i) \models \mathsf{G}\varphi \text{ iff } (T, j) \models \varphi \text{ for all } j \geq i$$

Example: HyperQLTL

There is a timepoint (common for all traces) where a is false in each trace. Not expressible in HyperLTL, but is in HyperQPTL.

$$\exists p \forall \pi\, \mathsf{G}(p \to \mathsf{XG}\neg p) \wedge \mathsf{F}(p \wedge \neg a_\pi)$$

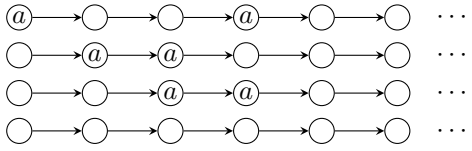


Example: TeamLTL

There is a timepoint (common for all traces) where a is false in each trace. Not expressible in HyperLTL, but is in HyperQPTL.

$$\exists p \forall \pi\, \mathsf{G}(p \to \mathsf{XG}\neg p) \wedge \mathsf{F}(p \wedge \neg a_\pi)$$

Expressible in synchronous TeamLTL: $\mathsf{F}\,\neg a$



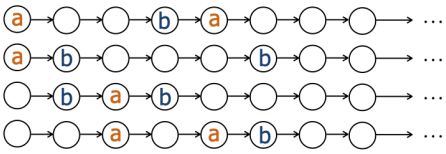
Examples: Disjunction in TeamLTL

A trace-set T satisfies $\varphi \vee \psi$ if it decomposed to sets T_φ and T_ψ satisfying φ and ψ.

$$(T,i) \models \varphi \vee \psi \text{ iff } (T_1,i) \models \varphi \text{ and } (T_2,i) \models \psi, \text{ for some } T_1 \cup T_2 = T$$
$$(T,i) \models \varphi \wedge \psi \text{ iff } (T,i) \models \varphi \text{ and } (T,i) \models \psi$$

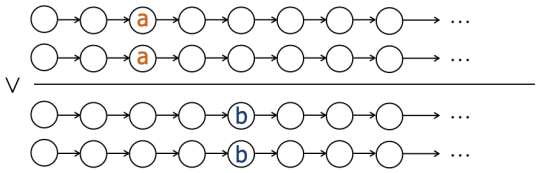
HyperLTL:

$\forall\pi.\forall\pi'.\ \mathrm{F}((a_\pi \wedge a_{\pi'}) \vee (b_\pi \wedge b_{\pi'}))$



TeamLTL:

$(\mathrm{F}\ a) \vee (\mathrm{F}\ b)$



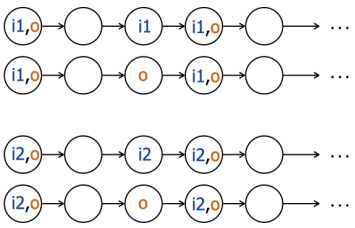
Examples: Dependence atom in TeamLTL

Dependence atom $\mathrm{dep}(p_1,\ldots,p_m,q)$ states that $p_1,\ldots,p_m$ functionally determine q:

$$(T,i) \models \mathrm{dep}(p_1,\ldots,p_m,q) \text{ iff } \forall t,t' \in T\Big(\bigwedge_{1\le j\le m} t[i](p_j) = t'[i](p_j)\Big) \Rightarrow (t[i](q) = t'[i](q))$$

$(\mathrm{G}\ \mathit{dep}(i1,o)) \vee (\mathrm{G}\ \mathit{dep}(i2,o))$

Nondeterministic dependence: "o either depends on i1 or on i2"



"whenever the traces agree on i1, they agree on o"

∨

"whenever the traces agree on i2, they agree on o"

Temporal team semantics

Definition 56. Temporal team is (T, i), where T a set of traces and $i \in \mathbb{N}$.

$$
\begin{array}{lll}
(T,i) \models p & \text{iff} & \forall t \in T : t[0](p) = 1 \\
(T,i) \models \neg p & \text{iff} & \forall t \ \in T : t[0](p) = 0 \\
(T,i) \models \phi \wedge \psi & \text{iff} & (T,i) \models \phi \text{ and } (T,i) \models \psi \\
(T,i) \models \phi \vee \psi & \text{iff} & (T_1,i) \models \phi \text{ and } (T_2,i) \models \psi, \text{ for some } T_1, T_2 \text{ s.t. } T_1 \cup T_2 = T \\
(T,i) \models \mathsf{X}\varphi & \text{iff} & (T, i+1) \models \varphi \\
(T,i) \models \phi \mathsf{U} \psi & \text{iff} & \exists k \geq i \text{ s.t. } (T,k) \models \psi \text{ and } \forall m : i \leq m < k \Rightarrow (T,m) \models \phi \\
(T,i) \models \phi \mathsf{W} \psi & \text{iff} & \forall k \geq i : (T,k) \models \phi \text{ or } \exists m \text{ s.t. } i \leq m \leq k \text{ and } (T,m) \models \psi
\end{array}
$$

As usual $\mathsf{F}\varphi \coloneqq (\top \mathsf{U} \varphi)$ and $\mathsf{G}\varphi \coloneqq (\varphi \mathsf{W} \bot)$. TeamLTL$(\Diamond, \subseteq)$ is the extension with the atoms and extra connectives in the brackets.

Generalised atoms and complete logics

Let B be a set of n-ary Boolean relations. We define the property $[\varphi_1, \ldots, \varphi_n]_B$ for an n-tuple $(\varphi_1, \ldots, \varphi_n)$ of LTL-formulae:

$$
(T,i) \models [\varphi_1, \ldots, \varphi_n]_B \quad \text{iff} \quad \{([\![\phi_1]\!]_{(t,i)}, \ldots, [\![\phi_n]\!]_{(t,i)}) \mid t \in T\} \in B.
$$

Theorem 57. TeamLTL$(\Diamond, \mathrm{NE}, \overset{1}{\mathsf{A}})$ *can express all* $[\varphi_1, \ldots, \varphi_n]_B$. TeamLTL$(\Diamond, \overset{1}{\mathsf{A}})$ *can express all* $[\varphi_1, \ldots, \varphi_n]_B$, *for downward closed* B.

- $(T,i) \models \mathrm{NE}$ iff $T \neq \emptyset$.
- $(T,i) \models \overset{1}{\mathsf{A}}\varphi$ iff $(\{t\}, i) \models \varphi$, for all $t \in T$.

Complexity results

Logic	Model Checking Result
TeamLTL without $\vee$	in PSPACE
k-coherent TeamLTL$(\sim)$	in EXPSPACE
left-flat TeamLTL$(\Diamond, \overset{1}{\mathsf{A}})$	in EXPSPACE
TeamLTL$(\subseteq, \Diamond)$	Σ^0_1-hard
TeamLTL$(\subseteq, \Diamond, \mathsf{A})$	Σ^1_1-hard
TeamLTL$(\sim)$	complete for third-order arithmetic

- k-coherence: $(T,i) \models \varphi$ iff $(S,i) \models \varphi$ for all $S \subseteq T$ s.t. $|S| \leq k$
- left-flatness: Restrict U and W syntactically to $(\overset{1}{\mathsf{A}}\varphi \mathsf{U} \psi)$ and $(\overset{1}{\mathsf{A}}\varphi \mathsf{W} \psi)$
- $\sim$ is contradictory negation and TeamLTL$(\sim)$ subsumes all the other logics

Source of Undecidability

Definition 58. A non-deterministic 3-counter machine M consists of a list I of n instructions that manipulate three counters C_l, C_m and C_r. All instructions are of the following forms:

- C_a^+ goto $\{j_1, j_2\}$, C_a^- goto $\{j_1, j_2\}$, if $C_a = 0$ goto $j_1$else goto j_2,

where $a \in \{l, m, r\}$, $0 \leq j_1, j_2 < n$.

- configuration: tuple (i, j, k, l), where $0 \leq i < n$ is the next instruction to be executed, and $j, k, l \in \mathbb{N}$ are the current values of the counters C_l, C_m and C_r.
- computation: infinite sequence of consecutive configurations starting from the initial configuration $(0, 0, 0, 0)$.
- computation b-recurring if the instruction labelled b occurs infinitely often in it.
- computation is lossy if the counter values can non-deterministically decrease

Undecidability results

Theorem 59 (Alur & Henzinger 1994, Schnoebelen 2010)**.** *Deciding whether a given non-deterministic 3-counter machine has a (lossy) b-recurring computation for a given b is (Σ_1^0-complete) Σ_1^1-complete.*

Theorem 60 ([Vir+21])**.** *Model checking for* TeamLTL$(\Diamond, \subseteq)$ *is Σ_0^1-hard. Model checking for* TeamLTL$(\Diamond, \subseteq, \mathsf{A})$ *is Σ_1^1-hard.*

Proof Idea:

- reduce existence of b-recurring computation of given 3-counter machine M and instruction label b to model checking problem of TeamLTL$(\Diamond, \subseteq, \mathsf{A})$
- TeamLTL$(\Diamond, \subseteq)$ suffices to enforce lossy computation
- $(T[i, \infty], 0)$ encodes the value of counters of the ith configuration the value of C_a is the cardinality of the set $\{t \in T[i, \infty] \mid t[0](c_a) = 1\}$

Model checking for TeamLTL$(\subseteq, \Diamond)$ is Σ_1^0-hard.

Proof. Given a set I of instructions of a 3-counter machine M, and an instruction label b, we construct a TeamLTL$(\subseteq, \Diamond)$-formula $\varphi_{I,b}$ and a Kripke structure $\mathfrak{K}_I$ such that

$$\big(\text{Traces}(\mathfrak{K}_I), 0\big) \models \varphi_{I,b} \quad \text{iff} \quad M \text{ has a } b\text{-recurring lossy computation.} \tag{5.1}$$

The Σ_1^0-hardness then follows since the construction is computable. □

Idea of the encoding
Put $n \coloneqq |I|$. A set T of traces using propositions $\{c_l, c_m, c_r, d, 0, \ldots, n-1\}$ encodes the sequence $(\vec{c}_j)_{j\in\mathbb{N}}$ of configurations, if for each $j \in \mathbb{N}$ and $\vec{c}_j = (i, v_l, v_m, v_r)$

- $t[j] \cap \{0, \ldots, n-1\} = \{i\}$, for all $t \in T$,
- $|\{t[j,\infty] \mid c_s \in t[j], t \in T\}| = v_s$, for each $s \in \{l, m, r\}$.

Hence, we use $T[j,\infty]$ to encode the configuration $\vec{c}_j$; the propositions $0, \ldots, n-1$ are used to encode the next instruction, and c_l, c_m, c_r, d are used to encode the values of the counters. The proposition d is a dummy proposition used to separate traces with identical postfixes with respect to c_l, c_m, and c_r.

Construction of the Kripke structure
The Kripke structure $\mathfrak{K}_I = (W, R, \eta, w_0)$ over the set of propositions $\{c_l, c_m, c_r, d, 0, \ldots, n-1\}$ is defined such that every possible sequence of configurations of M starting from $(0,0,0,0)$ can be encoded by some team $(T, 0)$, where $T \subseteq \mathrm{Traces}(\mathfrak{K}_I)$.

Construction of the formula
The connective $\vee_{\mathrm{L}}$ is a shorthand for the condition:

$$(T,i) \models \phi \vee_{\mathrm{L}} \psi \text{ iff } \exists T_1, T_2 \text{ s.t. } T_1 \neq \emptyset,\ T_1 \cup T_2 = T, (T_1, i) \models \phi \text{ and } (T_2, i) \models \psi.$$

The disjunction $\vee_{\mathrm{L}}$ can be defined using $\subseteq$, $\vee$ (see e.g., [Hel+19, Lemma 3.4]).

The formula

$$\theta_{b-\mathrm{rec}} \coloneqq \mathsf{GF}b$$

describes the b-recurrence condition of the computation.

The formula $\phi_{I,b}$ enforces that the configurations encoded by $T[i,\infty]$, $i \in \mathbb{N}$, encode an accepting computation of the counter machine; $\vee_{\mathrm{L}}$ guesses the computation.

$$\phi_{I,b} \coloneqq (\theta_{\mathrm{comp}} \wedge \theta_{b-\mathrm{rec}}) \vee_{\mathrm{L}} \top.$$

The formula θ_{comp} states that the encoded computation is legal.

Claim 61. *The claim (5.1) on page 49 holds.*

Proof. First, define

$$\text{singleton} \coloneqq \mathsf{G} \bigwedge_{a\in\mathrm{PROP}} (a \mathbin{⫽} \neg a), \qquad \mathrm{c_s}\text{-decrease} \coloneqq c_s \vee (\neg c_s \wedge \mathsf{X}\neg c_s), \text{ for } s \in \{l, m, r\}.$$

The intuitive idea behind the above formulae are as follows: A team satisfying the formula singleton contains at most a single trace with respect to the propositions in PROP. If a team (T, i) satisfies $\mathrm{c_s}$-decrease, then the number of traces in $T[i+1,\infty]$ satisfying c_s is less or equal to the number of traces in $T[i,\infty]$ satisfying c_s. In our encoding of counters this would mean that the value of the counter c in the configuration

$\vec{c}_{i+1}$ is less or equal to its value in the configuration $\vec{c}_i$. Thus $\mathrm{c_s}$-decrease will be handy below for encoding lossy computation.

Next, for each instruction label i, we define a formula θ_i describing the result of the execution of the instruction:

- For the instruction $i\colon C_l^+$ goto $\{j, j'\}$, define
  $\theta_i \coloneqq \mathsf{X}(j \mathbin{\text{⫽}} j') \wedge \big((\text{singleton} \wedge \neg c_l \wedge \mathsf{X} c_l) \vee \mathrm{c_l}\text{-decrease}\big) \wedge \mathrm{c_r}\text{-decrease} \wedge \mathrm{c_m}\text{-decrease}\,.$
- For the instruction $i\colon C_l^-$ goto $\{j, j'\}$, define
  $\theta_i \coloneqq \mathsf{X}(j \mathbin{\text{⫽}} j') \wedge \big((c_l \wedge \mathsf{X}\neg c_l) \vee_{\mathrm{L}} \mathrm{c_l}\text{-decrease}\big) \wedge \mathrm{c_r}\text{-decrease} \wedge \mathrm{c_m}\text{-decrease}\,.$
- For the instrunctions $i\colon C_s^+$ goto $\{j, j'\}$ and $i\colon C_s^-$ goto $\{j, j'\}$ with $s \in \{m, r\}$, the formulae θ_i are defined analogously with the indices l, m, and r permuted.
- For the instruction $i\colon$ if $C_s = 0$ goto j, else goto j', define
  $\theta_i \coloneqq \big(\mathsf{X}(\neg c_s \wedge j) \mathbin{\text{⫽}} (\top \subseteq c_s \wedge \mathsf{X} j')\big) \wedge \mathrm{c_l}\text{-decrease} \wedge \mathrm{c_m}\text{-decrease} \wedge \mathrm{c_r}\text{-decrease}\,.$

Finally, define $\theta_{\mathrm{comp}} \coloneqq \mathsf{G} \mathbin{\text{⫽}}_{i<n} (i \wedge \theta_i)$. We next describe the intuition of the above formulae. The left-most conjunct of θ_i for $i\colon C_l^+$ goto $\{j, j'\}$ expresses that after executing the instruction i, the label of the next instruction is either j or j'. The third and the fourth conjunct express that the values of counters C_r and C_m will not increase, but might decrease. The second conjunct expresses that the value of the counter C_l might increase by one, stay the same, or decrease. The meaning of θ_i for $i\colon C_l^-$ goto $\{j, j'\}$ is similar. Finally, the formula θ_i for $i\colon$ if $C_s = 0$ goto j, else goto j' expresses that, in the lossy execution of i, a) the values of the counters C_l, C_m, and C_r might decrease, but cannot increase, b) the next instruction is either j or j', c) if the next instruction is j then the value of the counter C_s, after the lossy execution, is 0, and d) if the next instruction is j' then the value of the counter C_s, before the lossy execution, was not 0.

Next, we define the Kripke structure $\mathfrak{K}_I = (W, R, \eta, w_0)$ over the set of propositions $\{c_l, c_m, c_r, d, 0, \ldots, n-1\}$. The structure is defined such that every possible sequence of configurations of M starting from $(0, 0, 0, 0)$ can be encoded by some team $(T, 0)$, where $T \subseteq \mathrm{Traces}(\mathfrak{K}_I)$. Define $W \coloneqq \{(i, j, k, t, l) \mid 0 \leq i < n \text{ and } j, k, t, l \in \{0, 1\}\}$, $w_0 \coloneqq (0, 0, 0, 0, 0)$, $R \coloneqq W \times W$, and η as the valuation such that $\eta\big((i, j, k, t, l)\big) \cap \{0, \ldots, n-1\} = i$,

- $c_l \in \eta\big((i, j, k, t, l)\big)$ if $j = 1$,
- $c_r \in \eta\big((i, j, k, t, l)\big)$ if $t = 1$,
- $c_m \in \eta\big((i, j, k, t, l)\big)$ if $k = 1$,
- and $d \in \eta\big((i, j, k, t, l)\big)$ if $l = 1$.

Assume first that M has a b-recurring lossy computation and let $(\vec{c}_j)_{j \in \mathbb{N}}$ be the related sequence of configurations of M. Let $T \subseteq \mathrm{Traces}(\mathfrak{K}_I)$ be a set of traces that encodes $(\vec{c}_j)_{j \in \mathbb{N}}$ in the way described above, and such that, for every $j \in \mathbb{N}$,

- if $\vec{c}_j = (i, v_l, v_m, v_r)$ and the instruction labelled i is of the form $i\colon C_s^+$ goto $\{j, j'\}$, then there is at most one $t \in T[j, \infty]$ such that $c_s \notin t[0]$ but $c_s \in t[1]$,

- if i is not of the form $i\colon C_s^+$ goto $\{j, j'\}$, then there is no $t \in T[j, \infty]$ such that $c_s \notin t[0]$ but $c_s \in t[1]$.

The aforementioned condition makes sure that the traces in T that encode the incrementation of counter values do not change erratically. Clearly, such a T always exists, given that $(\vec{c}_j)_{j\in\mathbb{N}}$ encodes a lossy computation. Furthermore, since T encodes $(\vec{c}_j)_{j\in\mathbb{N}}$ and the related b-recurrent lossy computation follows the instructions in I, we have that $(T, 0) \models \theta_{\text{comp}} \wedge \theta_{b-\text{rec}}$. Finally, as $\emptyset \neq T \subseteq \text{Traces}(\mathfrak{K}_I)$, $\big(\text{Traces}(\mathfrak{K}_I), 0\big) \models (\theta_{\text{comp}} \wedge \theta_{b-\text{rec}}) \vee_{\text{L}} \top$ follows.

Assume then that $\big(\text{Traces}(\mathfrak{K}_I), 0\big) \models \phi_{I,b}$. Hence there exists some nonempty subset T of $\text{Traces}(\mathfrak{K}_I)$ such that $(T, 0) \models \theta_{\text{comp}} \wedge \theta_{b-\text{rec}}$. It is now easy to construct a sequence $(\vec{c}_j)_{j\in\mathbb{N}}$ of configurations that encode a b-recurrent lossy computation for M; for each $j \in \mathbb{N}$, define $\vec{c}_j = (i, v_l, v_m, v_r)$ such that $\bigcup T[j] \cap \{0, \ldots, n-1\} = \{i\}$, and $|\{t[j, \infty] \mid c_s \in t[j], t \in T\}| = v_s$, for each $s \in \{l, m, r\}$. □

Conclusion of Lecture 5

- Introduction into Temporal Logics
- Hyperproperties and Temporal Team Semantics
- Undecidability of model checking of TeamLTL$(\oslash, \subseteq)$

Bibliography

[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.

[Ebb+13] Johannes Ebbing, Lauri Hella, Arne Meier, Julian-Steffen Müller, Jonni Virtema and Heribert Vollmer. 'Extended Modal Dependence Logic'. In: *Logic, Language, Information, and Computation - 20th International Workshop, WoLLIC 2013, Darmstadt, Germany, August 20-23, 2013. Proceedings*. Ed. by Leonid Libkin, Ulrich Kohlenbach and Ruy J. G. B. de Queiroz. Vol. 8071. Lecture Notes in Computer Science. Springer, 2013, pp. 126–137. DOI: 10.1007/978-3-642-39992-3_13. URL: https://doi.org/10.1007/978-3-642-39992-3%5C_13.

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

[Eij14] Jan van Eijck. 'Dynamic Epistemic Logics'. In: *Johan van Benthem on Logic and Information Dynamics*. Springer, 2014, pp. 175–202.

[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.

[Gal12] Pietro Galliani. 'Inclusion and exclusion dependencies in team semantics - On some logics of imperfect information'. In: *Ann. Pure Appl. Log.* 163.1 (2012), pp. 68–84. DOI: 10.1016/J.APAL.2011.08.005. URL: https://doi.org/10.1016/j.apal.2011.08.005.

[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.

[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.

[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.

[Hem96] Edith Hemaspaandra. 'The Price of Universality'. In: *Notre Dame Journal of Formal Logic* 37.2 (1996), pp. 174–203. DOI: 10.1305/ndjfl/1040046086.

[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).

[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.

[Vää07] Jouko A. Väänänen. *Dependence Logic - A New Approach to Independence Friendly Logic*. Vol. 70. London Mathematical Society student texts. Cambridge University Press, 2007.

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

[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.