Complexity and Expressivity of Propositional Logics with Team Semantics

ESSLLI 2024 course

Arne Meier[1] Jonni Virtema[2]

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

Version of 8th August 2024

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.

Lecture 1: Propositional Logics with Team Semantics

Literature: [YV17]

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”

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 $a, c \in A$

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

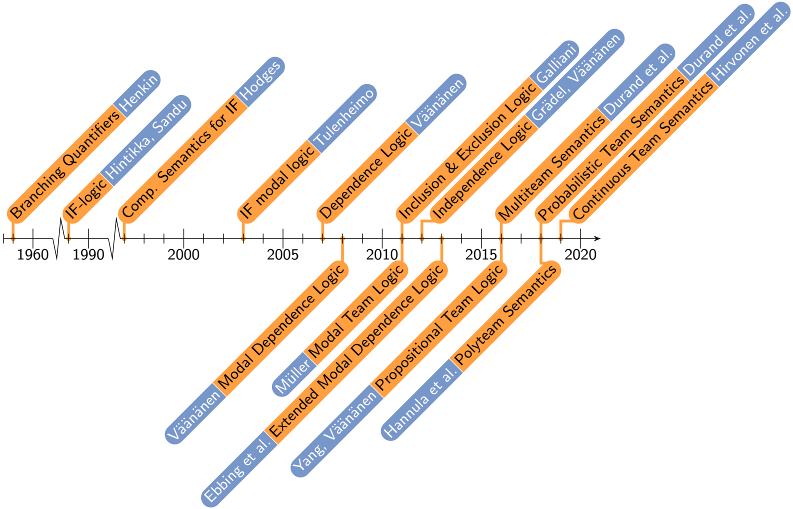
(In-)Dependence Logics: Independence-friendly logic (1989)

- 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

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
⋮	⋮	⋮	⋮

Primary key: docent, time

Dependence Logic: A Bit of Motivation

$\overbrace{\text{docent} \quad \text{time}}^{\text{Primary key}}$		room	lecture
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

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

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 $s\colon \mathrm{PROP} \to \{0,1\}$.
- Propositional Team Logic (PL):

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

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

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

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

Notation: $\mathrm{PL}[\mathrm{dep}]$ for propositional logic with dependence atoms.
Observations:

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

Team Semantics from the Database Perspective

propositions $\triangleq$ attributes

assignment $\triangleq$ entries

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
⋮	⋮	⋮	⋮

team $\triangleq$ table

Back to the Initial Example

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
⋮	⋮	⋮	⋮

$\underbrace{\text{docent} \quad \text{time}}_{\text{primary key}}$

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

Back to the Initial Example

$\overbrace{\text{docent} \quad \text{time}}^{\text{primary key}}$		room	lecture
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
$\vdots$	$\vdots$	$\vdots$	$\vdots$

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

How do you express this in $\mathrm{PL}[\mathrm{dep}]$?

Back to the Initial Example

$\overbrace{\text{docent} \quad \text{time}}^{\text{primary key}}$		room	lecture
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 $\mathrm{PL}[\mathrm{dep}]$?

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

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	$\subseteq$	$\perp$	$\mid$
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 $\mathrm{PL}[\mathrm{dep}]$

Lemma 3

$\mathrm{PL}[\mathrm{dep}]$ *is downward closed.*

Decision Problems

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

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

Theorem 4 ([Loh12, Theorem 4.13], Proof: Thursday)

$\mathrm{PL}[\mathrm{dep}]\text{-}\mathrm{MC}$ *is* NP-*complete.*

Satisfiability Does not Become Harder Than in the Classical Case

Theorem 5 ([Coo71; Lev73])

$\mathrm{PL}[\mathrm{dep}]\text{-}\mathrm{SAT}$ *is* NP*-complete.*

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 6 ([Hel+20, Cor. 3.6])

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

Theorem 7 ([Hel+19, Thm. 13])

$\mathrm{PL}[\subseteq]$-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

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

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

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

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 8 ([Han+18])

PL[$\perp$]-SAT *and* PL[$\perp$]-MC *are* NP-*complete.*

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 9 (by vanilla SAT, [Coo71; Lev73])

$\mathrm{PL}[|]$-*SAT is* NP-*complete.*

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

Theorem 10 ([Han19, Thm. 5.6, Thm. 6.1])

$\mathrm{PL[dep]\text{-}IMP}$ *is* coNEXPTIME$^{\mathrm{NP}}$*-complete.*

Conclusion of Lecture 1

- Team semantics
- Dependence Atoms
- Inclusion, Exclusion, Independence
- Complexity of Satisfiability of $\mathrm{PL}[\mathrm{dep}]$
- Properties of $\mathrm{PL}[\mathrm{dep}]$

Complexity and Expressivity of Propositional Logics with Team Semantics
Arne Meier, Jonni Virtema
6th of August

Lecture 2: Expressive power of team-based logics

Literature: [YV17; Hel+14]

How to characterise expressivity – Tarski's semantics

Definition 11

If φ is formula of propositional logic, with variables $p_1 \ldots, p_n$, 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 $p_1 \ldots, p_n$.

One can then ask, which Boolean functions can be expressed in propositional logic.

Definition 11

If φ is formula of propositional logic, with variables $p_1 \ldots, p_n$, 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 $p_1 \ldots, p_n$.

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).

Proposition 12

Every Boolean function can be defined in propositional logic.

How to characterise expressivity – Team semantics – definitions

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

Definition 13

We define

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

We then want to know, what are the families of teams that can be written as $\mathrm{Teams}(\varphi)$ by some formula φ.

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

Definition 13

We define

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

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 14

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 $t \in T$.

Properties of families of teams

Proposition 15

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.

Properties of families of teams

Proposition 15

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 $\mathcal{T} = \emptyset$. $\square$

Proposition 16

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.

Proposition 16

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 15, $\mathcal{T}$ is union closed and downward closed.

Proposition 16

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 15, $\mathcal{T}$ is union closed and downward closed. From union closure of $\mathcal{T}$ it follows that $\bigcup \mathcal{T} \in \mathcal{T}$.

Proposition 16

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 15, $\mathcal{T}$ 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 $T \in \mathcal{T}$. $\square$

How to characterise expressivity – Team semantics – results

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

Proposition 17

- *A family of teams defined by a PL-formula is flat.*
- $\mathrm{PL}[\mathrm{dep}]$*-definable team families are downward closed and include the empty team.*

How to characterise expressivity – Team semantics – results

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

Proposition 17

- *A family of teams defined by a PL-formula is flat.*
- $\mathrm{PL}[\mathrm{dep}]$*-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.

How to characterise expressivity – Team semantics – results

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

Proposition 17

- *A family of teams defined by a PL-formula is flat.*
- $\mathrm{PL}[\mathrm{dep}]$*-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.

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

How to characterise expressivity – Team semantics – results

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

Proposition 17

- *A family of teams defined by a PL-formula is flat.*
- $\mathrm{PL}[\mathrm{dep}]$*-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. $\square$

How to characterise expressivity – Team semantics – results

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

Proposition 17

- *A family of teams defined by a PL-formula is flat.*
- $\mathrm{PL}[\mathrm{dep}]$-*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. $\square$

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

Expressivity of PL with team semantics

Proposition 18

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

Proof.

Expressivity of PL with team semantics

Proposition 18

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 12.

Expressivity of PL with team semantics

Proposition 18

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 12. We will then define

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

and claim that $\mathcal{T} = \mathrm{Teams}(\Phi)$.

Expressivity of PL with team semantics

Proposition 18

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 12. We will then define

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

and claim that $\mathcal{T} = \mathrm{Teams}(\Phi)$. It is easy to check that $T \models \Phi$ if and only if $T \subseteq \bigcup \mathcal{T}$.

Expressivity of PL with team semantics

Proposition 18

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 12. We will then define

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

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 16, the latter holds if and only if $T \in \mathcal{T}$. $\square$

Expressivity of PL with team semantics

Proposition 18

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 12. We will then define

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

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 16, the latter holds if and only if $T \in \mathcal{T}$. $\square$

Theorem 19

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

Expressivity: the downward closed case

Let's consider an extension $\mathrm{PL}[\varovee]$ of PL with the so-called Boolean disjunction

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

Proposition 20

$\mathrm{PL}[\varovee]$ *is downward closed and has the empty team property.*

Expressivity: the downward closed case

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

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

Proposition 20

$\mathrm{PL}[\otimes]$ *is downward closed and has the empty team property.*

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

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

where $p^{\bot} := \neg p$ and $p^{\top} := p$.

Expressive power of $\mathrm{PL}[\varovee]$

Theorem 21

A family of teams is definable in $\mathrm{PL}[\varovee]$ *if and only if the family is downward closed and includes the empty team.*

Expressive power of $\mathrm{PL}[\varovee]$

Theorem 21

A family of teams is definable in $\mathrm{PL}[\varovee]$ *if and only if the family is downward closed and includes the empty team.*

Proof.

Let $\mathcal{T}$ be a family of teams with the aforementioned properties. Define

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

We claim that $\mathrm{Teams}(\Phi) = \mathcal{T}$.

Expressive power of $\mathrm{PL}[\varovee]$

Theorem 21

A family of teams is definable in $\mathrm{PL}[\varovee]$ *if and only if the family is downward closed and includes the empty team.*

Proof.

Let $\mathcal{T}$ be a family of teams with the aforementioned properties. Define

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

We claim that $\mathrm{Teams}(\Phi) = \mathcal{T}$. If $S \models \Phi$, there is some $T \in \mathcal{T}$ s.t. $S \models \bigvee_{s \in T} \varphi_s$.

Expressive power of $\mathrm{PL}[\varovee]$

Theorem 21

A family of teams is definable in $\mathrm{PL}[\varovee]$ *if and only if the family is downward closed and includes the empty team.*

Proof.

Let $\mathcal{T}$ be a family of teams with the aforementioned properties. Define

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

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.

Expressive power of $\mathrm{PL}[\varovee]$

Theorem 21

A family of teams is definable in $\mathrm{PL}[\varovee]$ if and only if the family is downward closed and includes the empty team.

Proof.

Let $\mathcal{T}$ be a family of teams with the aforementioned properties. Define

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

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 $S \models \Phi$. $\square$

Expressive power of $\mathrm{PL}[\otimes]$

Theorem 21

A family of teams is definable in $\mathrm{PL}[\otimes]$ if and only if the family is downward closed and includes the empty team.

Proof.

Let $\mathcal{T}$ be a family of teams with the aforementioned properties. Define

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

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 $S \models \Phi$. □

Can you make the formula a bit shorter?

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 $\theta_\Gamma \coloneqq \bigwedge_{\psi \in \Gamma} \psi \wedge \bigwedge_{\psi \in \Psi \setminus \Gamma} \neg\psi$,

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

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 $\theta_\Gamma \coloneqq \bigwedge_{\psi \in \Gamma} \psi \wedge \bigwedge_{\psi \in \Psi \setminus \Gamma} \neg\psi$,

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.

Types and characterising formulae

We define some auxiliary notation and formulae:

- $\mathrm{Type}_\Psi(s) := \{\varphi \in \Psi \mid s \models \varphi\}$, for a set of PL-formulae Ψ and an assignment s.
- For $\Gamma \subseteq \Psi$, define $\theta_\Gamma := \bigwedge_{\psi\in\Gamma} \psi \wedge \bigwedge_{\psi\in\Psi\setminus\Gamma} \neg\psi$,

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

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

Lemma 22

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 $T \models \bigvee\!\!\!\!\bigcirc\, \Psi$ and $\mathrm{Type}_\Psi(S) \subseteq \mathrm{Type}_\Psi(T)$, then $S \models \bigvee\!\!\!\!\bigcirc\, \Psi$.*

Case 1. follows by flatness of PL,

Types and characterising formulae

We define some auxiliary notation and formulae:

- $\mathrm{Type}_\Psi(s) := \{\varphi \in \Psi \mid s \models \varphi\}$, for a set of PL-formulae Ψ and an assignment s.
- For $\Gamma \subseteq \Psi$, define $\theta_\Gamma := \bigwedge_{\psi \in \Gamma} \psi \wedge \bigwedge_{\psi \in \Psi \setminus \Gamma} \neg\psi$,

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

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

Lemma 22

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

Case 1. follows by flatness of PL, and 2. uses 1. together with the definition of $\ovee$. Intuitively, it follows due to downward closure.

Expressive power of $\mathrm{PL}[\mathrm{dep}]$

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

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

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

Expressive power of $\mathrm{PL}[\mathrm{dep}]$

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

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

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

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

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

Lemma 23

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

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

Lemma 23

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

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

Proof.

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

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

Lemma 23

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

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

Proof.

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

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

Now given a team S we have

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

Theorem 24

$\mathrm{PL}[\varovee]$ *is equi-expressive with* $\mathrm{PL}[\mathrm{dep}]$.

Theorem 24

$\mathrm{PL}[\varovee]$ *is equi-expressive with* $\mathrm{PL}[\mathrm{dep}]$.

Proof.

$\mathrm{PL}[\varovee] \leq \mathrm{PL}[\mathrm{dep}]$ direction: Let $\varphi = \bigcirc\!\!\!\!\bigvee \Psi$ be a $\mathrm{PL}[\varovee]$-formula in a normal form, where $\Psi \subseteq \mathrm{PL}$. Define

$$\eta := \bigwedge_{T \notin \mathrm{Teams}(\varphi)} \xi_T, \text{ where } \xi_T \text{ is as in Lemma 23.}$$

Intuitively $S \models \eta$ iff no falsifying team of φ is completely subsumed by S.

Theorem 24

$\mathrm{PL}[\varovee]$ *is equi-expressive with* $\mathrm{PL}[\mathrm{dep}]$.

Proof.

$\mathrm{PL}[\varovee] \leq \mathrm{PL}[\mathrm{dep}]$ direction: Let $\varphi = \bigvee\!\!\!\!\bigcirc \Psi$ be a $\mathrm{PL}[\varovee]$-formula in a normal form, where $\Psi \subseteq \mathrm{PL}$. Define

$$\eta := \bigwedge_{T \notin \mathrm{Teams}(\varphi)} \xi_T, \text{ where } \xi_T \text{ is as in Lemma 23.}$$

Intuitively $S \models \eta$ iff no falsifying team of φ is completely subsumed by S.
By definition η is a $\mathrm{PL}[\mathrm{dep}]$-formula. To prove that $\mathrm{Teams}(\eta) = \mathrm{Teams}(\varphi)$, assume first that $S \in \mathrm{Teams}(\varphi)$, and consider any $T \notin \mathrm{Teams}(\varphi)$.

Theorem 24

$\mathrm{PL}[\varovee]$ *is equi-expressive with* $\mathrm{PL}[\mathrm{dep}]$.

Proof.

$\mathrm{PL}[\varovee] \leq \mathrm{PL}[\mathrm{dep}]$ direction: Let $\varphi = \bigovee \Psi$ be a $\mathrm{PL}[\varovee]$-formula in a normal form, where $\Psi \subseteq \mathrm{PL}$. Define

$$\eta := \bigwedge_{T \notin \mathrm{Teams}(\varphi)} \xi_T, \text{ where } \xi_T \text{ is as in Lemma 23.}$$

Intuitively $S \models \eta$ iff no falsifying team of φ is completely subsumed by S.
By definition η is a $\mathrm{PL}[\mathrm{dep}]$-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 22 that $\mathrm{Type}_\Psi(T) \not\subseteq \mathrm{Type}_\Psi(S)$. Hence by Lemma 23, $S \models \xi_T$. Thus $S \in \mathrm{Teams}(\eta)$.

Theorem 24

$\mathrm{PL}[\varovee]$ *is equi-expressive with* $\mathrm{PL}[\mathrm{dep}]$.

Proof.

$\mathrm{PL}[\varovee] \leq \mathrm{PL}[\mathrm{dep}]$ direction: Let $\varphi = \bigovee \Psi$ be a $\mathrm{PL}[\varovee]$-formula in a normal form, where $\Psi \subseteq \mathrm{PL}$. Define

$$\eta := \bigwedge_{T \notin \mathrm{Teams}(\varphi)} \xi_T, \text{ where } \xi_T \text{ is as in Lemma 23.}$$

Intuitively $S \models \eta$ iff no falsifying team of φ is completely subsumed by S.
By definition η is a $\mathrm{PL}[\mathrm{dep}]$-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 22 that $\mathrm{Type}_\Psi(T) \not\subseteq \mathrm{Type}_\Psi(S)$. Hence by Lemma 23, $S \models \xi_T$. 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 23 that $S \not\models \xi_S$. Thus $S \notin \mathrm{Teams}(\eta)$. $\square$

Definition 25

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,

Dimensions of team families

Definition 25

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 26

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 $\dim(\varphi) \leq \mathrm{Dim}(\varphi)$.

Estimates for the upper dimension

Lemma 27

We have the following upper dimension estimates for $\varphi, \psi \in \mathrm{PL}[\varovee]$:

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. $\mathrm{Dim}(\varphi \varovee \psi) \leq \mathrm{Dim}(\varphi) + \mathrm{Dim}(\psi)$.

Proof.

We omit the cases for (1) and (3), since (1) is trivial, and (3) is analogous to (2).

Estimates for the upper dimension

Lemma 27

We have the following upper dimension estimates for $\varphi, \psi \in \mathrm{PL}[\varovee]$:

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

Proof.

We omit the cases for (1) and (3), since (1) is trivial, and (3) is analogous to (2). We defer the proof of (2) to the lecture notes.
Case (4): For the Boolean disjunction, it holds that

$$M(\varphi \varovee \psi) \subseteq M(\varphi) \cup M(\psi)$$

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

What are dimensions good for?

Proposition 28

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

Proposition 29

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

What are dimensions good for?

Proposition 28

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

Proposition 29

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

Theorem 30

Let $\varphi \in \mathrm{PL}[\varovee]$ *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 28, $\mathrm{Dim}(\varphi) = \mathrm{Dim}(\mathrm{dep}(p_1, \ldots, p_n, q)) = 2^{2^n}$. Thus $2^{2^n} \leq 2^{\mathrm{occ}_{\varovee}(\varphi)}$ by Prop. 29, implying $2^n \leq \mathrm{occ}_{\varovee}(\varphi)$. Hence φ has at least 2^n Boolean disjunctions. $\square$

What are dimensions good for?

Proposition 28

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

Proposition 29

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

Theorem 30

Let $\varphi \in \mathrm{PL}[\varovee]$ *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 28, $\mathrm{Dim}(\varphi) = \mathrm{Dim}(\mathrm{dep}(p_1, \ldots, p_n, q)) = 2^{2^n}$. Thus $2^{2^n} \leq 2^{\mathrm{occ}_{\varovee}(\varphi)}$ by Prop. 29, implying $2^n \leq \mathrm{occ}_{\varovee}(\varphi)$. Hence φ has at least 2^n Boolean disjunctions. $\square$

Thus, any translation from $\mathrm{PL}[\mathrm{dep}]$ to $\mathrm{PL}[\varovee]$ leads to an exponential blow-up.

Expressivity of propositional inclusion logic

Theorem 31

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 $\mathrm{PL}[\ovee]$ and its equivalence with $\mathrm{PL}[\mathrm{dep}]$. The result was first shown in [HS15]. $\square$

Conclusion of Lecture 2

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

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

Lecture 3: Inclusion Logic

Literature: [Hel+19; Hel+20]

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 32

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

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

Validity in Inclusion Logic is Hard

Theorem 33

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

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.

Foundations: Monotone circuit value problem

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

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

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

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

Proposition 34 ([Gol77])

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

Model-Checking for Inclusion Logic

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

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

Model-Checking for Inclusion Logic

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

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

Ideas:

Lower bound: reduce from MCVP

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

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

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

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

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

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

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

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

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

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

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

After skipping some technicalities we arrive at

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

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

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:

$$\psi_{\mathrm{out}=1} \coloneqq p_\top \subseteq p_0,$$

$$\psi_\wedge \coloneqq \bigwedge\{p_i \subseteq p_j \mid (g_j, g_i) \in E \text{ and } \alpha(g_i) = \wedge\},$$

$$\psi_\vee \coloneqq \bigwedge\{p_k \subseteq p_{k=i\vee j} \mid i < j, (g_i, g_k) \in E, (g_j, g_k) \in E, \text{ and } \alpha(g_k) = \vee\}$$

Encoding MCVP: the final puzzle pieces

More truth about the team:

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

Encoding MCVP: the final puzzle pieces

More truth about the team:

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

Now we claim that

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

Encoding MCVP: the final puzzle pieces

More truth about the team:

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

Now we claim that

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

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

Encoding MCVP: the final puzzle pieces

More truth about the team:

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

Now we claim that

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

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

Computing a Maximum Satisfying Team

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

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

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

Computing a Maximum Satisfying Team

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

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

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

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

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

Important properties:

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

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

Important properties:

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

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

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

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

Important properties:

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

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

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

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

Membership in P: more formally

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

Membership in P: more formally

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

bottum-up part (odd i):

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

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

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 $T \models \varphi$

Satisfiability for Inclusion Logic

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

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

Short ideas:

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

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

Satisfiability for Inclusion Logic

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

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

Short ideas:

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

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

Persistent subsets

Definition 38

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

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

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

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

PER *is* P*-complete.*

A succinct variant of PER

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

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

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

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

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

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

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

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

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

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

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

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

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

Encoding $\mathrm{S\text{-}PER}$ into $\mathrm{PL}[\subseteq]$

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

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

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

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

Encoding $\mathrm{S\text{-}PER}$ into $\mathrm{PL}[\subseteq]$

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

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

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

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

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

Encoding persistency into the formula

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

Encoding persistency into the formula

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

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

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

Encoding persistency into the formula

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

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

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

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

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

"$\Rightarrow$" of the claim

Conclusion of Lecture 3

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

Lecture 4: Complexity of propositional dependence logic and beyond

Literature: [Vir17; Han+18]

Canonical complete problems: SAT and QBF

SAT [Coo71]		QBF (Stockmeyer and Meyer, 1973)	
Input:	Boolean formula θ	Input:	Quantified Boolean formula $\phi := Q_1 p_1 \dots Q_n p_n \theta$
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 41 ([Loh12, Theorem 4.13])

$\mathrm{PL}[\mathrm{dep}]\text{-}\mathrm{MC}$ *is* NP-*complete.*

Proof ideas:

Membership: Use nondeterminism for splitjunctions.

Hardness: reduce from 3SAT.

Membership in NP

NP Lower Bound

Canonical complete problems: DQBF

DQBF (Peterson, Reif, Azhar, 2001)	
Input:	Dependency Quantified Boolean formula $\phi := \forall p_1 \ldots \forall p_m \exists q_1 \ldots \exists q_n \theta$ and constraints $\vec{c}_1, \ldots, \vec{c}_n$
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.

Canonical complete problems: DQBF

DQBF (Peterson, Reif, Azhar, 2001)	
Input:	Dependency Quantified Boolean formula $\phi := \forall p_1 \dots \forall p_m \exists q_1 \dots \exists q_n \theta$ 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 $\vec{c}_i$.

Canonical complete problems: DQBF

DQBF (Peterson, Reif, Azhar, 2001)	
Input:	Dependency Quantified Boolean formula $\phi := \forall p_1 \dots \forall p_m \exists q_1 \dots \exists q_n \theta$ 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 $\mathrm{dep}(\vec{c}_1, q_1) \wedge \dots \wedge \mathrm{dep}(\vec{c}_n, q_n) \wedge \theta$!

The validity problem for PD is in NEXPTIME

If $D \subseteq \mathrm{PROP}$, we denote by 2^D the set of all assignments $s\colon D \to \{0,1\}$.

Lemma 42

A $\mathrm{PL}[\mathrm{dep}]$-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. $\square$

The validity problem for PD is in NEXPTIME

If $D \subseteq \text{PROP}$, we denote by 2^D the set of all assignments $s\colon D \to \{0,1\}$.

Lemma 42

A $\mathrm{PL}[\mathrm{dep}]$*-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. $\square$

Lemma 43

The validity problem for $\mathrm{PL}[\mathrm{dep}]$ *is in NEXPTIME.*

Proof.

Let $\varphi \in \mathrm{PL}[\mathrm{dep}]$ whose variables are in D. By Lemma 42, 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.

The validity problem for PD is in NEXPTIME

If $D \subseteq \text{PROP}$, we denote by 2^D the set of all assignments $s\colon D \to \{0,1\}$.

Lemma 42

A $\mathrm{PL}[\mathrm{dep}]$*-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. $\square$

Lemma 43

The validity problem for $\mathrm{PL}[\mathrm{dep}]$ *is in NEXPTIME.*

Proof.

Let $\varphi \in \mathrm{PL}[\mathrm{dep}]$ whose variables are in D. By Lemma 42, 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 41, 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 $|\varphi|$. $\square$

The validity problem for PD is NEXPTIME-hard

We will associate each DQBF-formula μ with a corresponding $\mathrm{PL}[\mathrm{dep}]$-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., $D_\mu := \{p_1, \dots, p_n, q_1, \dots, q_k\}$.

The validity problem for PD is NEXPTIME-hard

We will associate each DQBF-formula μ with a corresponding $\mathrm{PL}[\mathrm{dep}]$-formula φ_μ. Let

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

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

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

We will show that μ is true if and only if the $\mathrm{PL}[\mathrm{dep}]$-formula φ_μ is valid.

The validity problem for PD is NEXPTIME-hard

We will associate each DQBF-formula μ with a corresponding $\mathrm{PL}[\mathrm{dep}]$-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., $D_\mu := \{p_1, \dots, p_n, q_1, \dots, q_k\}$. 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

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

We will show that μ is true if and only if the $\mathrm{PL}[\mathrm{dep}]$-formula φ_μ is valid. By Lemma 42, it suffices to show that μ is valid if and only if $2^{D_\mu} \models \varphi_\mu$.

The validity problem for PD is NEXPTIME-hard

We will associate each DQBF-formula μ with a corresponding $\mathrm{PL}[\mathrm{dep}]$-formula φ_μ. Let

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

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

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

We will show that μ is true if and only if the $\mathrm{PL}[\mathrm{dep}]$-formula φ_μ is valid. By Lemma 42, 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 $\mathrm{PL}[\mathrm{dep}]$ is NEXPTIME-hard.

A logic to rule them all

The extension of PL with the contradictory negation $\mathrm{PL}[\sim]$

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

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

A logic to rule them all

The extension of PL with the contradictory negation $\mathrm{PL}[\sim]$

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

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

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

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

Also dependence/inclusion/independence atoms can be expressed in $\mathrm{PL}[\sim]$ with polynomial blow up [LV19].

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

PTIME Reductions Between Validity and Satisfiability

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

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

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

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

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 44

- 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 AEXPTIME(poly)) by allowing queries to polynomially many oracles.

Complexity of $\mathrm{PL}[\sim]$

Theorem 45

$\mathrm{SAT}(\mathrm{PL}[\sim])$ *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. $\square$

Complexity of $\mathrm{PL}[\sim]$

Theorem 45

$\mathrm{SAT}(\mathrm{PL}[\sim])$ *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. $\square$

Corollary 46

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

Complexity of $\mathrm{PL}[\sim]$

Theorem 45

$\mathrm{SAT}(\mathrm{PL}[\sim])$ *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 46

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

Theorem 47

$\mathrm{MC}(\mathrm{PL}[\sim])$ *is* PSPACE*-complete*

Complexity Results

Logic	SAT	VAL	MC
PL	NP [0]	coNP [0]	NC[1] [1]
$\mathrm{PL}[\mathrm{dep}]$	NP [3]	NEXPTIME [4]	NP [2]
$\mathrm{PL}[\perp_{\mathrm{c}}]$	NP [7]	in $\mathrm{coNEXPTIME}^{\mathrm{NP}}$ [7]	NP [7]
$\mathrm{PL}[\subseteq]$	EXP [5]	coNP [7]	in P [6]
$\mathrm{PL}[\sim]$	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.
- $\mathrm{SAT}(\mathrm{PL}[\mathrm{dep}])$ and $\mathrm{MC}(\mathrm{PL}[\mathrm{dep}])$ are NP-complete.
- $\mathrm{VAL}(\mathrm{PL}[\mathrm{dep}])$ is NEXPTIME-complete.
- $\mathrm{SAT}(\mathrm{PL}[\sim])$ and $\mathrm{VAL}(\mathrm{PL}[\sim])$ are AEXPTIME(poly)-complete.
- $\mathrm{MC}(\mathrm{PL}[\sim])$ are PSPACE-complete.

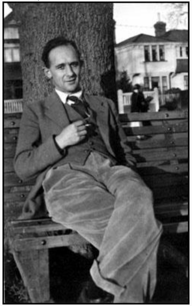
Complexity and Expressivity of Propositional Logics with Team Semantics
Arne Meier, Jonni Virtema
9th of August

Lecture 5: Recent Trends: Hyperproperties

Literature: [Vir+21; Gut+22]

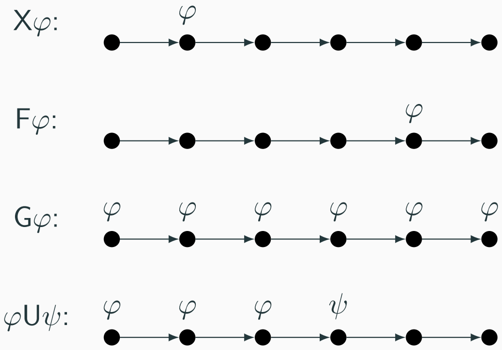
Temporal Logic

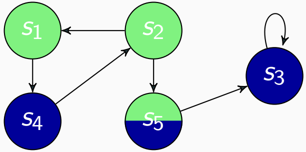
- Dates back to Arthur Norman Prior (1914–1969)
- New modalities neXt, Until, Future, Global
- and quantifiers: All paths, Exists a path
- ‘From Philosophical to Industrial Logics’ [Var09]



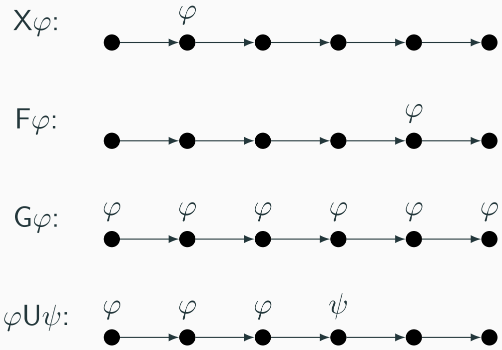
Arthur N. Prior
(1914–1969)

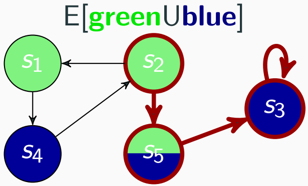
Temporal Logic: Semantics by Example



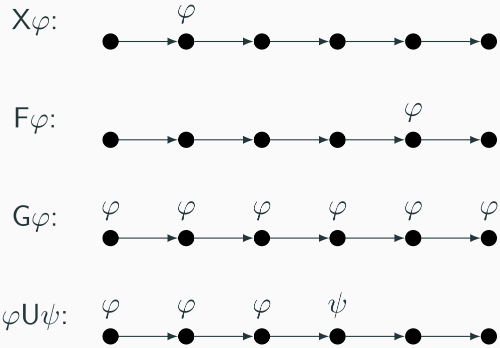


Temporal Logic: Semantics by Example

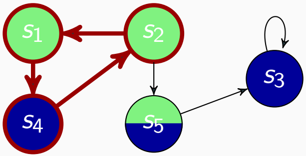




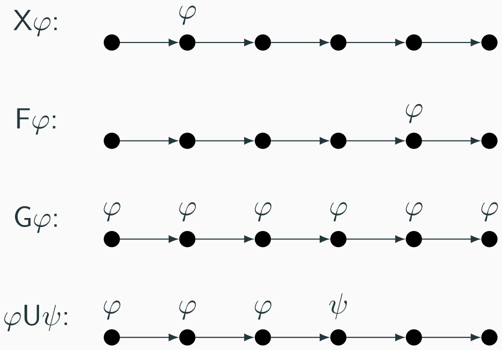
Temporal Logic: Semantics by Example

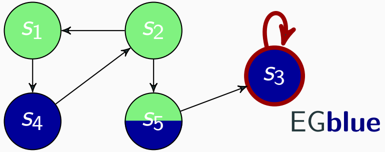


$\mathrm{EF}(\textbf{blue} \wedge \mathrm{AX}\textbf{green})$

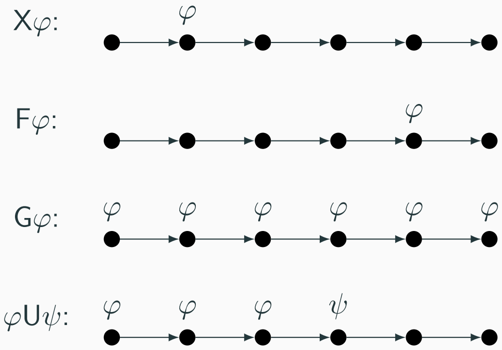


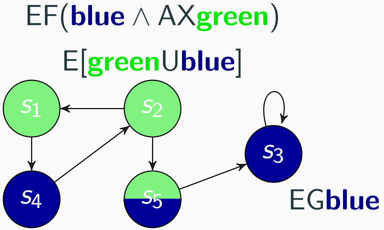
Temporal Logic: Semantics by Example



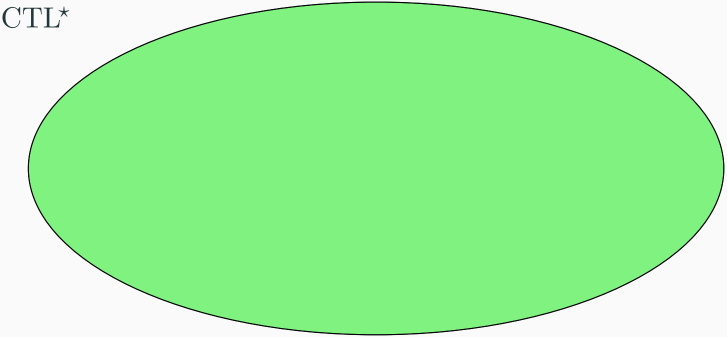


Temporal Logic: Semantics by Example





A Temporal Lanscape of Logics

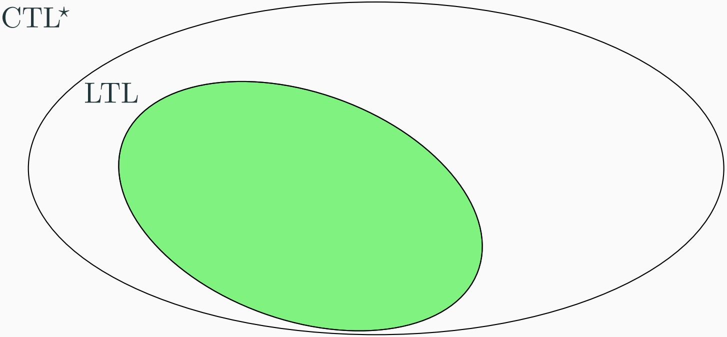


$\mathrm{CTL}^{\star}$: Syntax

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

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

A Temporal Lanscape of Logics

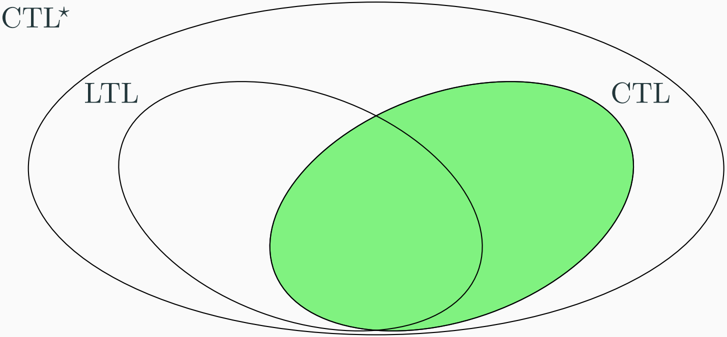


LTL: Formulae Eφ with

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

where $x \in$ PROP.

A Temporal Lanscape of Logics



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 $\mathrm{CTL}^{\star}$, 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 Eψ is a state formula.

State and Path Formulae

Regarding the formulae of $\mathrm{CTL}^{\star}$, 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 Eψ 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 Xφ, [φUψ].

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

Kripke Frames and Paths

Definition 48

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).

Kripke Frames and Paths

Definition 48

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 49

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 $V \colon \mathrm{PROP} \to \mathfrak{P}(W)$ is a labeling function.

Definition 50

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 $\pi_i := w_i w_{i+1} \ldots$, $\pi[i] := w_i$.

Formal Semantics of Temporal Logics

Definition 51

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

$$\mathcal{M}, w \models p \text{ iff } w \in V(p),$$
$$\mathcal{M}, w \models \neg\varphi \text{ iff } \mathcal{M}, w \not\models \varphi,$$
$$\mathcal{M}, w \models \varphi \wedge \psi \text{ iff } \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi,$$
$$\mathcal{M}, w \models \mathsf{E}\varphi \text{ iff there is a path } \pi \text{ with } \pi[0] = w \text{ we have that } \mathcal{M}, \pi \models \varphi,$$

Definition 51

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

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

Remaning Operators by Short Cuts

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

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

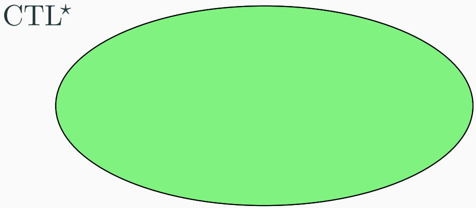
Two Important Problems and their Complexities

Satisfiability ($\mathrm{CTL}^\star$-SAT)

Given: $\mathrm{CTL}^\star$-formula φ.

Question: Is φ satisfiable?

Complexity: EEXP-complete [KV85; EJ99]



	SAT	MC
$\mathrm{CTL}^\star$	EEXP	
LTL		
CTL		

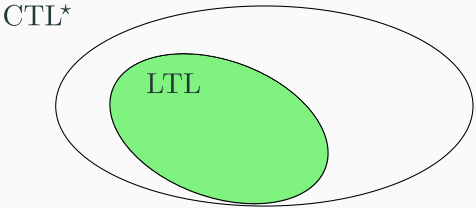
Two Important Problems and their Complexities

Satisfiability (LTL-SAT)

Given: LTL-formula φ.

Question: Is φ satisfiable?

Complexity: PSPACE-complete [SC85]



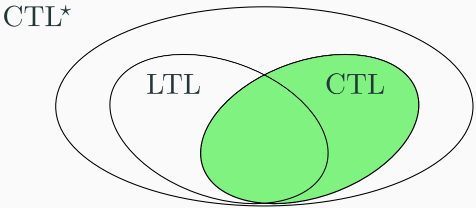
	SAT	MC
CTL*	EEXP	
LTL	PSPACE	
CTL		

Satisfiability (CTL-SAT)

Given: CTL-formula φ.

Question: Is φ satisfiable?

Complexity: EXP-complete [FL79; Pra80]



	SAT	MC
CTL*	EEXP	
LTL	PSPACE	
CTL	EXP	

Two Important Problems and their Complexities

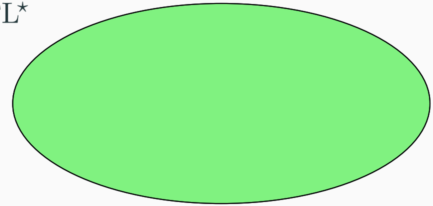
Model Checking (CTL*-MC)

Given: CTL*-formula φ, model $\mathcal{M}$.

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

Complexity: PSPACE-complete [CES86]

CTL*



	SAT	MC
CTL*	EEXP	PSPACE
LTL	PSPACE	
CTL	EXP	

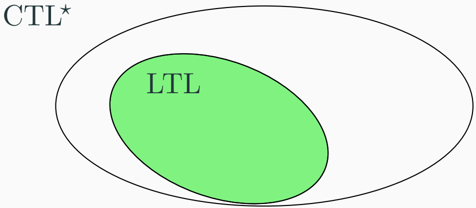
Two Important Problems and their Complexities

Model Checking (LTL-MC)

Given: LTL-formula φ, model $\mathcal{M}$.

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

Complexity: PSPACE-complete [CES86]



	SAT	MC
CTL*	EEXP	PSPACE
LTL	PSPACE	PSPACE
CTL	EXP	

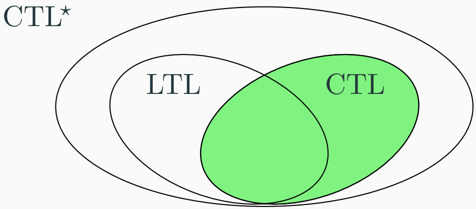
Two Important Problems and their Complexities

Model Checking (CTL-MC)

Given: CTL-formula φ, model $\mathcal{M}$.

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

Complexity: P-complete [CES86; Sch02]



	SAT	MC
CTL*	EEXP	PSPACE
LTL	PSPACE	PSPACE
CTL	EXP	P

Examples for Interesting Properties in Temporal Logics

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

Examples for Interesting Properties in Temporal Logics

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

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

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

Examples for Interesting Properties in Temporal Logics

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

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

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

$$\mathrm{AF}p$$

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

Examples for Interesting Properties in Temporal Logics

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

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

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

$$\mathrm{AF}p$$

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

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

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.

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:

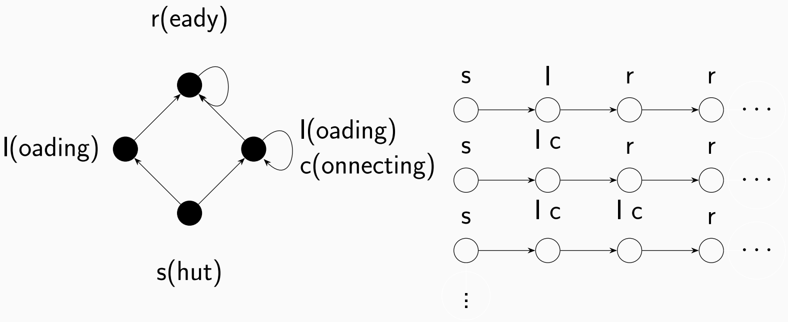
- Check whether a given system satisfies a given specification.

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

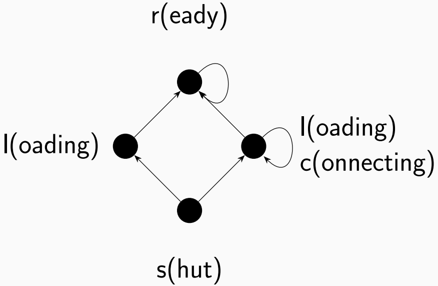
- LTL, QPTL, CTL, etc. vs. HyperLTL, HyperQPTL, HyperCTL, etc.
  are prominent logics for traceproperties vs. hyperproperties of systems
  - Traceproperty: Each request is eventually granted (properties of traces)
  - Hyperproperty: Non-inference (values of public outputs do not leak information about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

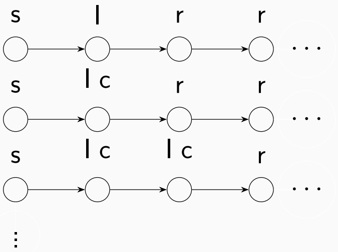
Properties of quantification based hyperproperties

- 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.

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)





Hyperlogic via team semantics

- Temporal logics with team semantics express hyperproperties.
- Purely modal logic & well suited for properties of unbounded number of traces.

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.

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

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

Semantics of TeamLTL

Temporal team semantics is universal and synchronous

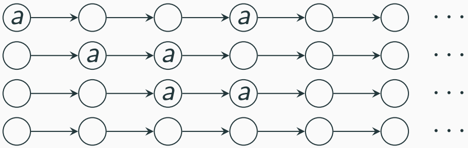
$$(T, i) \models p \text{ iff } \forall t \in T : t[i](p) = 1 \qquad (T, i) \models \neg p \text{ iff } \forall t \in T : t[i](p) = 0$$

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

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.

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

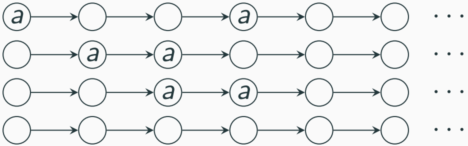


Example: TeamLTL

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

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

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



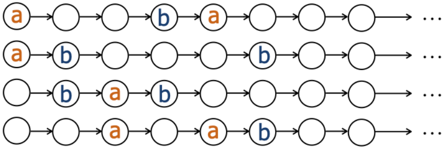
Examples: Disjunction in TeamLTL

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

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

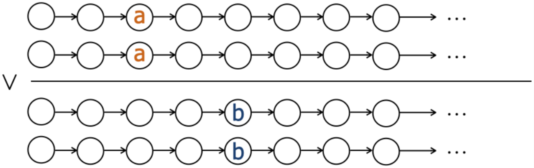
HyperLTL:

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



TeamLTL:

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



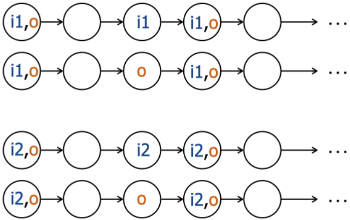
Examples: Dependence atom in TeamLTL

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

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

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

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



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

$\vee$

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

Temporal team semantics

Definition 52

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

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

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:

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

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:

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

Theorem 53

$\mathrm{TeamLTL}(\varovee, \mathrm{NE}, \overset{1}{\mathsf{A}})$ *can express all* $[\varphi_1, \ldots, \varphi_n]_B$.

$\mathrm{TeamLTL}(\varovee, \overset{1}{\mathsf{A}})$ *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
$\mathrm{TeamLTL}$ without $\vee$	in PSPACE
k-coherent $\mathrm{TeamLTL}(\sim)$	in EXPSPACE
left-flat $\mathrm{TeamLTL}(\varovee, \overset{1}{\mathsf{A}})$	in EXPSPACE
$\mathrm{TeamLTL}(\subseteq, \varovee)$	Σ^0_1-hard
$\mathrm{TeamLTL}(\subseteq, \varovee, \mathsf{A})$	Σ^1_1-hard
$\mathrm{TeamLTL}(\sim)$	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 $\mathrm{TeamLTL}(\sim)$ subsumes all the other logics

Source of Undecidability

Definition 54

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

Source of Undecidability

Definition 54

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 $\{j_1, j_2\}$, $\quad$ C_a^- goto $\{j_1, j_2\}$, $\quad$ 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 55 (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^0_1*-complete*) Σ^1_1*-complete.*

Undecidability results

Theorem 55 (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^0_1-complete) Σ^1_1-complete.

Theorem 56 ([Vir+21])

Model checking for $\mathrm{TeamLTL}(\varovee, \subseteq)$ *is* Σ^1_0*-hard.*
Model checking for $\mathrm{TeamLTL}(\varovee, \subseteq, \mathsf{A})$ *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 $\mathrm{TeamLTL}(\varovee, \subseteq, \mathsf{A})$
- $\mathrm{TeamLTL}(\varovee, \subseteq)$ 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 $\mathrm{TeamLTL}(\subseteq, \varovee)$ is Σ^0_1-hard.

Proof.

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

$$(\mathrm{Traces}(\mathfrak{K}_I), 0) \models \varphi_{I,b} \quad \text{iff} \quad M \text{ has a } b\text{-recurring lossy computation.} \tag{1}$$

The Σ^0_1-hardness then follows since the construction is computable. $\square$

Idea of the encoding

Put $n := |I|$. 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 $\{c_l, c_m, c_r, d, 0, \dots, n-1\}$ 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]).

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

$$\theta_{b-\mathrm{rec}} := \mathsf{GF}b$$

describes the b-recurrence condition of the computation.

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

$$\theta_{b-\mathrm{rec}} := \mathsf{GF}b$$

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.

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

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

Expressing legality of computation

Define

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

Expressing legality of computation

Define

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

For the instruction $i\colon\ C_l^+$ goto $\{j, j'\}$, define

$$\theta_i := \mathsf{X}(j \otimes j') \wedge \big((\text{singleton} \wedge \neg c_l \wedge \mathsf{X}c_l) \vee \mathsf{c_l}\text{-decrease}\big) \wedge \mathsf{c_r}\text{-decrease} \wedge \mathsf{c_m}\text{-decrease}\,.$$

Expressing legality of computation

Define

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

For the instruction $i\colon\ C_l^+$ goto $\{j, j'\}$, define

$$\theta_i := \mathsf{X}(j \oslash j') \wedge \big((\text{singleton} \wedge \neg c_l \wedge \mathsf{X} c_l) \vee \mathsf{c_l}\text{-decrease}\,\big) \wedge \mathsf{c_r}\text{-decrease} \wedge \mathsf{c_m}\text{-decrease}\,.$$

For the instruction $i\colon$ if $C_s = 0$ goto j, else goto j', define

$$\theta_i := \big(\mathsf{X}(\neg c_s \wedge j) \oslash (\top \subseteq c_s \wedge \mathsf{X} j')\big) \wedge \mathsf{c_l}\text{-decrease} \wedge \mathsf{c_m}\text{-decrease} \wedge \mathsf{c_r}\text{-decrease}\,.$$

Expressing legality of computation

Define

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

For the instruction $i\colon C_l^+$ goto $\{j, j'\}$, define

$$\theta_i \coloneqq \mathsf{X}(j \varovee j') \wedge \big((\text{singleton} \wedge \neg c_l \wedge \mathsf{X} c_l) \vee \mathsf{c_l}\text{-decrease}\big) \wedge \mathsf{c_r}\text{-decrease} \wedge \mathsf{c_m}\text{-decrease}\,.$$

For the instruction $i\colon$ if $C_s = 0$ goto j, else goto j', define

$$\theta_i \coloneqq \big(\mathsf{X}(\neg c_s \wedge j) \varovee (\top \subseteq c_s \wedge \mathsf{X} j')\big) \wedge \mathsf{c_l}\text{-decrease} \wedge \mathsf{c_m}\text{-decrease} \wedge \mathsf{c_r}\text{-decrease}\,.$$

Finally, define $\theta_{\mathrm{comp}} \coloneqq \mathsf{G} \bigvee_{i<n} (i \wedge \theta_i)$.

Conclusion of Lecture 5

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

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

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

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

Bibliography ii

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

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

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

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

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

Bibliography iii

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

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

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

[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.

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

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

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

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

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

Bibliography vi

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

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

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

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

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

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

[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.