Descriptive complexity for counting complexity classes

Martín Muñoz

PUC Chile - IMFD

Joint work with
Marcelo Arenas and Cristian Riveros

Descriptive complexity has been very fruitful
in connecting **logics** with **computational complexity**

Descriptive complexity has been very fruitful
in connecting **logics** with **computational complexity**

$$\mathrm{NP} \quad \equiv \quad \exists\mathsf{SO}$$

Descriptive complexity has been very fruitful in connecting **logics** with **computational complexity**

$$\begin{array}{ccc}
\mathrm{NP} & \equiv & \exists\mathsf{SO} \\
\textsc{co}\mathrm{NP} & \equiv & \forall\mathsf{SO} \\
\mathrm{P} & \equiv & \mathsf{LFP}_{\leq} \\
\mathrm{NL} & \equiv & \mathsf{TC}_{\leq} \\
\mathrm{AC}_0 & \equiv & \mathsf{FO} + \mathtt{Bit} \\
\mathrm{PSPACE} & \equiv & \mathsf{PFP}_{\leq} \\
\vdots & \vdots & \vdots
\end{array}$$

Descriptive complexity has been very fruitful in connecting **logics** with **computational complexity**

$$
\begin{array}{ccc}
\mathrm{NP} & \equiv & \exists\mathsf{SO} \\
\textsc{co}\mathrm{NP} & \equiv & \forall\mathsf{SO} \\
\mathrm{P} & \equiv & \mathsf{LFP}_{\leq} \\
\mathrm{NL} & \equiv & \mathsf{TC}_{\leq} \\
\mathrm{AC}_0 & \equiv & \mathsf{FO} + \mathtt{Bit} \\
\mathrm{PSPACE} & \equiv & \mathsf{PFP}_{\leq} \\
\vdots & \vdots & \vdots
\end{array}
$$

Many **applications** in diverse areas like:

1. Computational complexity and logics.
2. Database management systems.
3. Verification systems.

. . . but computational complexity

is not only about `true` or `false`

. . . but computational complexity
is not only about `true` or `false`

One would like to study the **complexity** of problems like:

> *"How many valuations satisfies my boolean formula?"*

...but computational complexity is not only about `true` or `false`

One would like to study the **complexity** of problems like:

"How many valuations satisfies my boolean formula?"

"How many simple paths are connecting two vertices in my graph?"

...but computational complexity is not only about `true` or `false`

$$\#\mathrm{P}$$

$$\textsc{SpanP}$$

$$\mathrm{FP}$$

$$\#\mathrm{L}$$

$$\#\mathrm{PSPACE}$$

$$\vdots$$

...but computational complexity is not only about `true` or `false`

$$\text{Counting complexity classes} \begin{cases} \#\mathrm{P} \\ \textsc{SpanP} \\ \mathrm{FP} \\ \#\mathrm{L} \\ \#\mathrm{PSPACE} \\ \vdots \end{cases}$$

. . . but computational complexity is not only about `true` or `false`

$$\text{Counting complexity classes} \left\{ \begin{array}{ccc} \#\mathrm{P} & \equiv & ? \\ \textsc{SpanP} & \equiv & ? \\ \mathrm{FP} & \equiv & ? \\ \#\mathrm{L} & \equiv & ? \\ \#\mathrm{PSPACE} & \equiv & ? \\ \vdots & \vdots & \vdots \end{array} \right.$$

How can we describe these **counting** classes with logic?

In this paper, we propose to use weighted logics for descriptive complexity of counting classes

In this paper, we propose to use weighted logics for descriptive complexity of counting classes

We propose to use:

Quantitative Second Order Logics (QSO) $\quad = \quad$ Weighted Logics over $\mathbb{N}$

In this paper, we propose to use weighted logics for descriptive complexity of counting classes

We propose to use:

Quantitative Second Order Logics (QSO) $=$ Weighted Logics over $\mathbb{N}$

Specifically, our contributions are:

In this paper, we propose to use weighted logics for descriptive complexity of counting classes

We propose to use:

Quantitative Second Order Logics (QSO) = Weighted Logics over $\mathbb{N}$

Specifically, our contributions are:

1. We show that QSO **captures** many counting complexity classes.
   - $\#\mathrm{P}$, $\textsc{SpanP}$, FP, $\#\mathrm{PSPACE}$, $\textsc{MinP}$, $\textsc{MaxP}$, $\ldots$

In this paper, we propose to use weighted logics for descriptive complexity of counting classes

We propose to use:

Quantitative Second Order Logics (QSO) = Weighted Logics over $\mathbb{N}$

Specifically, our contributions are:

1. We show that QSO **captures** many counting complexity classes.
   - $\#\mathrm{P}$, $\textsc{SpanP}$, FP, $\#\mathrm{PSPACE}$, $\textsc{MinP}$, $\textsc{MaxP}$, $\ldots$
2. We use QSO to find classes below $\#\mathrm{P}$
   that have good **tractability** and **closure** properties.

In this paper, we propose to use weighted logics for descriptive complexity of counting classes

We propose to use:

Quantitative Second Order Logics (QSO) = Weighted Logics over $\mathbb{N}$

Specifically, our contributions are:

1. We show that QSO **captures** many counting complexity classes.
   - $\#\mathrm{P}$, $\textsc{SpanP}$, FP, $\#\mathrm{PSPACE}$, $\textsc{MinP}$, $\textsc{MaxP}$, $\ldots$
2. We use QSO to find classes below $\#\mathrm{P}$ that have good **tractability** and **closure** properties.
3. We show how to define **quantitative recursion** over QSO in order to capture classes below FP.

Outline

Quantitative second order logic

QSO vs counting complexity

Below and beyond

Outline

Quantitative second order logic

QSO vs counting complexity

Below and beyond

Some notation and restrictions

Given a relational signature $\mathbf{R} = \{R_1, \ldots, R_k, <\}$,
we consider **finite ordered structures** over $\mathbf{R}$ of the form:

$$\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \ldots, R_k^{\mathfrak{A}}, <^{\mathfrak{A}})$$

where A is the domain and $<^{\mathfrak{A}}$ is a **linear order** over A.

Some notation and restrictions

Given a relational signature $\mathbf{R} = \{R_1, \ldots, R_k, <\}$,
we consider **finite ordered structures** over $\mathbf{R}$ of the form:

$$\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \ldots, R_k^{\mathfrak{A}}, <^{\mathfrak{A}})$$

where A is the domain and $<^{\mathfrak{A}}$ is a **linear order** over A.

Let $\textsc{Struct}(\mathbf{R})$ be the set of all finite ordered structures over $\mathbf{R}$.

Some notation and restrictions

Given a relational signature $\mathbf{R} = \{R_1, \ldots, R_k, <\}$,
we consider **finite ordered structures** over $\mathbf{R}$ of the form:

$$\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \ldots, R_k^{\mathfrak{A}}, <^{\mathfrak{A}})$$

where A is the domain and $<^{\mathfrak{A}}$ is a **linear order** over A.

Let $\textsc{Struct}(\mathbf{R})$ be the set of all finite ordered structures over $\mathbf{R}$.

We consider formulas of **Second Order logic** over $\mathbf{R}$ of the form:

$$\varphi \;:=\; \texttt{True} \mid x = y \mid R(\bar{u}) \mid X(\bar{v}) \mid \neg\varphi \mid (\varphi \vee \varphi) \mid \exists x.\, \varphi \mid \exists X.\, \varphi$$

where $R \in \mathbf{R}$ and x and X are a first and second order variable, respectively.

Some notation and restrictions

Given a relational signature $\mathbf{R} = \{R_1, \ldots, R_k, <\}$,
we consider **finite ordered structures** over $\mathbf{R}$ of the form:

$$\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \ldots, R_k^{\mathfrak{A}}, <^{\mathfrak{A}})$$

where A is the domain and $<^{\mathfrak{A}}$ is a **linear order** over A.

Let $\textsc{Struct}(\mathbf{R})$ be the set of all finite ordered structures over $\mathbf{R}$.

We consider formulas of **Second Order logic** over $\mathbf{R}$ of the form:

$$\varphi \ := \ \texttt{True} \ \mid \ x = y \ \mid \ R(\bar{u}) \ \mid \ X(\bar{v}) \ \mid \neg\varphi \ \mid \ (\varphi \vee \varphi) \ \mid \ \exists x.\ \varphi \ \mid \ \exists X.\ \varphi$$

where $R \in \mathbf{R}$ and x and X are a first and second order variable, respectively.

The semantics of a second order formula is defined as usual.

Syntax of Quantitative Second Order logic

Syntax of Quantitative Second Order logic

Definition

A QSO-formula α over **R** is given by the following **syntax**:

$$\alpha \ := \ \varphi \in \mathsf{SO} \ \mid \ s \ \mid \ (\alpha + \alpha) \ \mid \ (\alpha \cdot \alpha) \ \mid \Sigma x.\,\alpha \ \mid \ \Pi x.\,\alpha \ \mid \ \Sigma X.\,\alpha \ \mid \ \Pi X.\,\alpha$$

where φ is a (boolean) second order formula and $s \in \mathbb{N}$.

Syntax of Quantitative Second Order logic

Definition

A QSO-formula α over **R** is given by the following **syntax**:

$$\alpha \;:=\; \varphi \in \mathsf{SO} \;\mid\; s \;\mid\; (\alpha + \alpha) \;\mid\; (\alpha \cdot \alpha) \;\mid \Sigma x.\,\alpha \;\mid\; \Pi x.\,\alpha \;\mid\; \Sigma X.\,\alpha \;\mid\; \Pi X.\,\alpha$$

where φ is a (boolean) second order formula and $s \in \mathbb{N}$.

Example

Let $\mathbf{R} = \{E(\cdot,\cdot), <\}$ where E encodes an edge relation.

$$\alpha \;:=\; \Sigma x.\Sigma y.\Sigma z.\big(E(x,y) \wedge E(y,z) \wedge E(z,x) \wedge x < y \wedge y < z\big)$$

Syntax of Quantitative Second Order logic

Definition

A QSO-formula α over $\mathbf{R}$ is given by the following **syntax**:

$$\alpha \ := \ \varphi \in \mathsf{SO} \ \mid \ s \ \mid \ (\alpha + \alpha) \ \mid \ (\alpha \cdot \alpha) \ \mid \Sigma x.\, \alpha \ \mid \ \Pi x.\, \alpha \ \mid \ \Sigma X.\, \alpha \ \mid \ \Pi X.\, \alpha$$

where φ is a (boolean) second order formula and $s \in \mathbb{N}$.

Example

Let $\mathbf{R} = \{E(\cdot,\cdot), <\}$ where E encodes an edge relation.

$$\alpha \ := \ \Sigma x.\, \Sigma y.\, \Sigma z.\, \Big(\underbrace{E(x,y) \wedge E(y,z) \wedge E(z,x) \wedge x < y \wedge y < z}_{\text{SO formula } \varphi} \Big)$$

Semantics of Quantitative Second Order logic

Semantics of Quantitative Second Order logic

Given a QSO-formula α, $\mathfrak{A} \in \textsc{Struct}(\mathbf{R})$ and a var. assignment $v : \mathbf{X} \to A$ we define the **semantics** $[\![\alpha]\!] : \textsc{Struct}(\mathbf{R}) \to \mathbb{N}$ recursively as follow:

Semantics of Quantitative Second Order logic

Given a QSO-formula α, $\mathfrak{A} \in \textsc{Struct}(\mathbf{R})$ and a var. assignment $v : \mathbf{X} \to A$ we define the **semantics** $[\![\alpha]\!] : \textsc{Struct}(\mathbf{R}) \to \mathbb{N}$ recursively as follow:

$$[\![\varphi]\!](\mathfrak{A}, v) = \begin{cases} 1 & \text{if } (\mathfrak{A}, v) \models \varphi \\ 0 & \text{otherwise} \end{cases}$$

Semantics of Quantitative Second Order logic

Given a QSO-formula α, $\mathfrak{A} \in \textsc{Struct}(\mathbf{R})$ and a var. assignment $v : \mathbf{X} \to A$ we define the **semantics** $[\![\alpha]\!] : \textsc{Struct}(\mathbf{R}) \to \mathbb{N}$ recursively as follow:

$$
\begin{aligned}
[\![\varphi]\!](\mathfrak{A}, v) &= \begin{cases} 1 & \text{if } (\mathfrak{A}, v) \models \varphi \\ 0 & \text{otherwise} \end{cases} \\
[\![s]\!](\mathfrak{A}, v) &= s
\end{aligned}
$$

Semantics of Quantitative Second Order logic

Given a QSO-formula α, $\mathfrak{A} \in \textsc{Struct}(\mathbf{R})$ and a var. assignment $v : \mathbf{X} \to A$ we define the **semantics** $[\![\alpha]\!] : \textsc{Struct}(\mathbf{R}) \to \mathbb{N}$ recursively as follow:

$$
\begin{aligned}
[\![\varphi]\!](\mathfrak{A}, v) &= \begin{cases} 1 & \text{if } (\mathfrak{A}, v) \models \varphi \\ 0 & \text{otherwise} \end{cases} \\
[\![s]\!](\mathfrak{A}, v) &= s \\
[\![\alpha_1 + \alpha_2]\!](\mathfrak{A}, v) &= [\![\alpha_1]\!](\mathfrak{A}, v) + [\![\alpha_2]\!](\mathfrak{A}, v) \\
[\![\alpha_1 \cdot \alpha_2]\!](\mathfrak{A}, v) &= [\![\alpha_1]\!](\mathfrak{A}, v) \cdot [\![\alpha_2]\!](\mathfrak{A}, v)
\end{aligned}
$$

Semantics of Quantitative Second Order logic

Given a QSO-formula α, $\mathfrak{A} \in \textsc{Struct}(\mathbf{R})$ and a var. assignment $v : \mathbf{X} \to A$ we define the **semantics** $[\![\alpha]\!] : \textsc{Struct}(\mathbf{R}) \to \mathbb{N}$ recursively as follow:

$$
\begin{aligned}
[\![\varphi]\!](\mathfrak{A}, v) &= \begin{cases} 1 & \text{if } (\mathfrak{A}, v) \models \varphi \\ 0 & \text{otherwise} \end{cases} \\
[\![s]\!](\mathfrak{A}, v) &= s \\
[\![\alpha_1 + \alpha_2]\!](\mathfrak{A}, v) &= [\![\alpha_1]\!](\mathfrak{A}, v) + [\![\alpha_2]\!](\mathfrak{A}, v) \\
[\![\alpha_1 \cdot \alpha_2]\!](\mathfrak{A}, v) &= [\![\alpha_1]\!](\mathfrak{A}, v) \cdot [\![\alpha_2]\!](\mathfrak{A}, v) \\
[\![\Sigma x.\, \alpha]\!](\mathfrak{A}, v) &= \sum_{a \in A} [\![\alpha]\!](\mathfrak{A}, v[a/x]) \\
[\![\Pi x.\, \alpha]\!](\mathfrak{A}, v) &= \prod_{a \in A} [\![\alpha]\!](\mathfrak{A}, v[a/x])
\end{aligned}
$$

Semantics of Quantitative Second Order logic

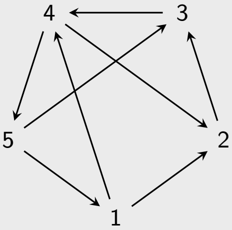
Given a QSO-formula α, $\mathfrak{A} \in \textsc{Struct}(\mathbf{R})$ and a var. assignment $v : \mathbf{X} \to A$ we define the **semantics** $[\![\alpha]\!] : \textsc{Struct}(\mathbf{R}) \to \mathbb{N}$ recursively as follow:

$$
\begin{aligned}
[\![\varphi]\!](\mathfrak{A}, v) &= \begin{cases} 1 & \text{if } (\mathfrak{A}, v) \models \varphi \\ 0 & \text{otherwise} \end{cases} \\
[\![s]\!](\mathfrak{A}, v) &= s \\
[\![\alpha_1 + \alpha_2]\!](\mathfrak{A}, v) &= [\![\alpha_1]\!](\mathfrak{A}, v) + [\![\alpha_2]\!](\mathfrak{A}, v) \\
[\![\alpha_1 \cdot \alpha_2]\!](\mathfrak{A}, v) &= [\![\alpha_1]\!](\mathfrak{A}, v) \cdot [\![\alpha_2]\!](\mathfrak{A}, v) \\
[\![\Sigma x.\, \alpha]\!](\mathfrak{A}, v) &= \sum_{a \in A} [\![\alpha]\!](\mathfrak{A}, v[a/x]) \\
[\![\Pi x.\, \alpha]\!](\mathfrak{A}, v) &= \prod_{a \in A} [\![\alpha]\!](\mathfrak{A}, v[a/x]) \\
[\![\Sigma X.\, \alpha]\!](\mathfrak{A}, v) &= \sum_{C \subseteq A^{\mathrm{arity}(X)}} [\![\alpha]\!](\mathfrak{A}, v[C/X]) \\
[\![\Pi X.\, \alpha]\!](\mathfrak{A}, v) &= \prod_{C \subseteq A^{\mathrm{arity}(X)}} [\![\alpha]\!](\mathfrak{A}, v[C/X])
\end{aligned}
$$

Semantics of Quantitative Second Order logic

Example (counting the triangles in a graph)

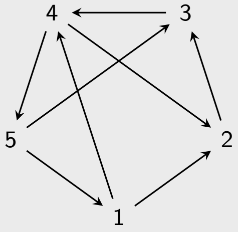
$\mathfrak{A}:$



Semantics of Quantitative Second Order logic

Example (counting the triangles in a graph)

$\mathfrak{A}$:

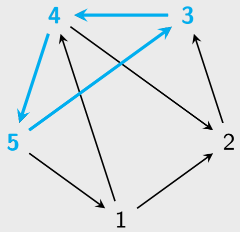


$$\text{triangle}(x,y,z) \;\coloneqq\; E(x,y) \wedge E(y,z) \wedge E(z,x) \wedge x < y \wedge y < z$$

Semantics of Quantitative Second Order logic

Example (counting the triangles in a graph)

$\mathfrak{A}$:

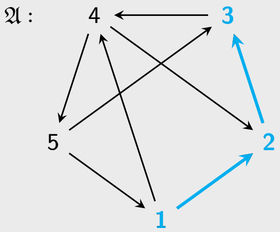


$$\text{triangle}(x, y, z) \;:=\; E(x, y) \wedge E(y, z) \wedge E(z, x) \wedge x < y \wedge y < z$$

$$[\![\text{triangle}]\!](\mathfrak{A}, 3, 4, 5) \;=\; 1$$

Semantics of Quantitative Second Order logic

Example (counting the triangles in a graph)



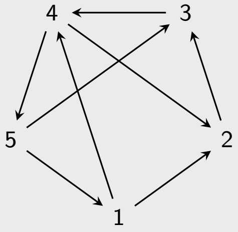
$$\text{triangle}(x, y, z) \;:=\; E(x, y) \wedge E(y, z) \wedge E(z, x) \wedge x < y \wedge y < z$$

$$[\![\text{triangle}]\!](\mathfrak{A}, 3, 4, 5) \;=\; 1 \qquad [\![\text{triangle}]\!](\mathfrak{A}, 1, 2, 3) \;=\; 0$$

Semantics of Quantitative Second Order logic

Example (counting the triangles in a graph)

$\mathfrak{A}:$



$$\text{triangle}(x, y, z) \ := \ E(x, y) \wedge E(y, z) \wedge E(z, x) \wedge x < y \wedge y < z$$

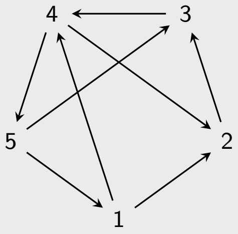
$$[\![\text{triangle}]\!](\mathfrak{A}, 3, 4, 5) \ = \ 1 \qquad [\![\text{triangle}]\!](\mathfrak{A}, 1, 2, 3) \ = \ 0$$

$$\alpha \ := \ \Sigma x. \Sigma y. \Sigma z. \ \text{triangle}(x, y, z)$$

Semantics of Quantitative Second Order logic

Example (counting the triangles in a graph)

$\mathfrak{A}:$



$$\text{triangle}(x,y,z) \;:=\; E(x,y) \wedge E(y,z) \wedge E(z,x) \wedge x < y \wedge y < z$$

$$[\![\text{triangle}]\!](\mathfrak{A},3,4,5) = 1 \qquad [\![\text{triangle}]\!](\mathfrak{A},1,2,3) = 0$$

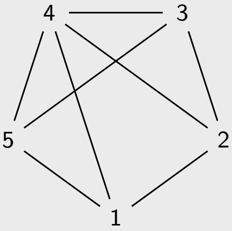
$$\alpha \;:=\; \Sigma x.\,\Sigma y.\,\Sigma z.\ \text{triangle}(x,y,z)$$

$$[\![\alpha]\!](\mathfrak{A}) = 3$$

Semantics of Quantitative Second Order logic

Example (counting the number of cliques in a graph)

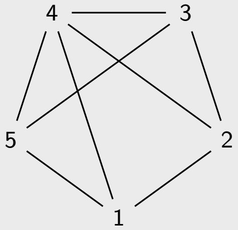
$\mathfrak{A}:$



Semantics of Quantitative Second Order logic

Example (counting the number of cliques in a graph)

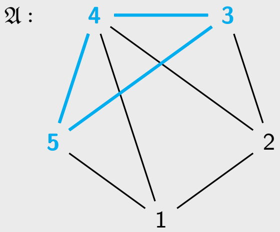
$\mathfrak{A}$:



$$\text{clique}(X) \coloneqq \forall x.\, \forall y.\, (X(x) \wedge X(y) \wedge x \neq y) \rightarrow E(x, y)$$

Semantics of Quantitative Second Order logic

Example (counting the number of cliques in a graph)

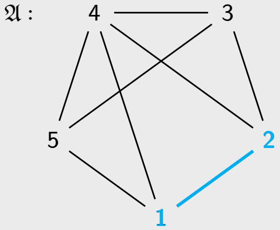


$$\text{clique}(X) \coloneqq \forall x.\, \forall y.\, (X(x) \wedge X(y) \wedge x \neq y) \rightarrow E(x, y)$$

$$[\![\text{clique}]\!](\mathfrak{A}, \{3, 4, 5\}) \;=\; 1$$

Semantics of Quantitative Second Order logic

Example (counting the number of cliques in a graph)

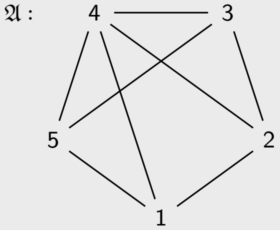


$$\text{clique}(X) \coloneqq \forall x.\, \forall y.\, (X(x) \wedge X(y) \wedge x \neq y) \rightarrow E(x,y)$$

$$[\![\text{clique}]\!](\mathfrak{A}, \{3,4,5\}) \;=\; 1 \qquad [\![\text{clique}]\!](\mathfrak{A}, \{1,2\}) \;=\; 1$$

Semantics of Quantitative Second Order logic

Example (counting the number of cliques in a graph)



$$\text{clique}(X) \coloneqq \forall x.\, \forall y.\, (X(x) \land X(y) \land x \neq y) \rightarrow E(x,y)$$

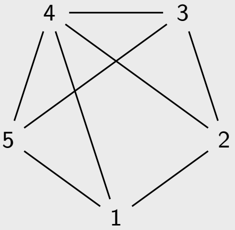
$$[\![\text{clique}]\!](\mathfrak{A}, \{3,4,5\}) = 1 \qquad [\![\text{clique}]\!](\mathfrak{A}, \{1,2\}) = 1$$

$$\alpha \coloneqq \Sigma X.\, \text{clique}(X)$$

Semantics of Quantitative Second Order logic

Example (counting the number of cliques in a graph)

$\mathfrak{A}:$



$$\mathsf{clique}(X) \coloneqq \forall x.\, \forall y.\, (X(x) \wedge X(y) \wedge x \neq y) \rightarrow E(x,y)$$

$$[\![\mathsf{clique}]\!](\mathfrak{A}, \{3,4,5\}) = 1 \qquad [\![\mathsf{clique}]\!](\mathfrak{A}, \{1,2\}) = 1$$

$$\alpha \coloneqq \Sigma X.\, \mathsf{clique}(X)$$

$$[\![\alpha]\!](\mathfrak{A}) = 18$$

Subfragments and extentions of QSO

$$\alpha \;:=\; \varphi \in \mathsf{SO} \;\mid\; s \;\mid\; (\alpha + \alpha) \;\mid\; (\alpha \cdot \alpha) \;\mid \Sigma x.\,\alpha \;\mid\; \Pi x.\,\alpha \;\mid\; \Sigma X.\,\alpha \;\mid\; \Pi X.\,\alpha$$

Subfragments and extentions of QSO

$$\alpha \;:=\; \varphi \in \mathsf{SO} \;\mid\; s \;\mid\; (\alpha + \alpha) \;\mid\; (\alpha \cdot \alpha) \;\mid \Sigma x.\,\alpha \;\mid\; \Pi x.\,\alpha \;\mid\; \Sigma X.\,\alpha \;\mid\; \Pi X.\,\alpha$$

$$\mathsf{QSO} \;=\; \underbrace{\mathsf{QSO}}_{\alpha}(\overbrace{\mathsf{SO}}^{\varphi})$$

Subfragments and extentions of QSO

$$\alpha \;:=\; \varphi \in \mathsf{SO} \;\mid\; s \;\mid\; (\alpha + \alpha) \;\mid\; (\alpha \cdot \alpha) \;\mid \Sigma x.\, \alpha \;\mid\; \Pi x.\, \alpha \;\mid\; \Sigma X.\, \alpha \;\mid\; \Pi X.\, \alpha$$

$$\mathsf{QSO} \;=\; \underbrace{\mathsf{QSO}}_{\alpha}(\overbrace{\mathsf{SO}}^{\varphi})$$

We can restrict or extend the language of φ:

QSO(FO) $:=$ φ is restricted to **FO logic**.

Subfragments and extentions of QSO

$$\alpha \;:=\; \varphi \in \mathsf{SO} \;\mid\; s \;\mid\; (\alpha + \alpha) \;\mid\; (\alpha \cdot \alpha) \;\mid \Sigma x.\,\alpha \;\mid\; \Pi x.\,\alpha \;\mid\; \Sigma X.\,\alpha \;\mid\; \Pi X.\,\alpha$$

$$\mathsf{QSO} \;=\; \underbrace{\mathsf{QSO}}_{\alpha}(\overbrace{\mathsf{SO}}^{\varphi})$$

We can restrict or extend the language of φ:

$\mathsf{QSO(FO)}$:= φ is restricted to **FO logic**.
$\mathsf{QSO(LFP)}$:= φ is restricted to **LFP logic**.

Subfragments and extentions of QSO

$$\alpha \; := \; \varphi \in \mathsf{SO} \;\mid\; s \;\mid\; (\alpha + \alpha) \;\mid\; (\alpha \cdot \alpha) \;\mid \Sigma x.\,\alpha \;\mid\; \Pi x.\,\alpha \;\mid\; \Sigma X.\,\alpha \;\mid\; \Pi X.\,\alpha$$

$$\mathsf{QSO} \; = \; \underbrace{\mathsf{QSO}}_{\alpha}(\overbrace{\mathsf{SO}}^{\varphi})$$

We can restrict or extend the language of φ:

$\mathsf{QSO(FO)}$ $:=$ φ is restricted to **FO logic**.
$\mathsf{QSO(LFP)}$ $:=$ φ is restricted to **LFP logic**.

We can restrict or extend the language of α:

$\mathsf{QFO(SO)}$ $:=$ α is restricted to **first order operators** (i.e. $s, +, \cdot, \Sigma x.\,, \Pi x.\,$).

Subfragments and extentions of QSO

$$\alpha \ := \ \varphi \in \mathsf{SO} \ | \ s \ | \ (\alpha + \alpha) \ | \ (\alpha \cdot \alpha) \ | \Sigma x.\, \alpha \ | \ \Pi x.\, \alpha \ | \ \Sigma X.\, \alpha \ | \ \Pi X.\, \alpha$$

$$\mathsf{QSO} \ = \ \underbrace{\mathsf{QSO}}_{\alpha}(\overbrace{\mathsf{SO}}^{\varphi})$$

We can restrict or extend the language of φ:

$\mathsf{QSO(FO)} \quad := \quad \varphi$ is restricted to **FO logic**.
$\mathsf{QSO(LFP)} \quad := \quad \varphi$ is restricted to **LFP logic**.

We can restrict or extend the language of α:

$\mathsf{QFO(SO)} \quad := \quad \alpha$ is restricted to **first order operators** (i.e. $s, +, \cdot, \Sigma x., \Pi x.$).
$\Sigma\mathsf{QSO(SO)} \quad := \quad \alpha$ is restricted to **sum operators** (i.e. $s, +, \Sigma x., \Sigma X.$)

Subfragments and extentions of QSO

$$\alpha \ := \ \varphi \in \mathsf{SO} \ \mid \ s \ \mid \ (\alpha + \alpha) \ \mid \ (\alpha \cdot \alpha) \ \mid \Sigma x.\, \alpha \ \mid \ \Pi x.\, \alpha \ \mid \ \Sigma X.\, \alpha \ \mid \ \Pi X.\, \alpha$$

$$\mathsf{QSO} \ = \ \underbrace{\mathsf{QSO}}_{\alpha}(\overbrace{\mathsf{SO}}^{\varphi})$$

We can restrict or extend the language of φ:

$$\begin{array}{rcl} \mathsf{QSO(FO)} & := & \varphi \text{ is restricted to } \textbf{FO logic}. \\ \mathsf{QSO(LFP)} & := & \varphi \text{ is restricted to } \textbf{LFP logic}. \end{array}$$

We can restrict or extend the language of α:

$$\begin{array}{rcl} \mathsf{QFO(SO)} & := & \alpha \text{ is restricted to } \textbf{first order operators} \text{ (i.e. } s, +, \cdot, \Sigma x., \Pi x.). \\ \Sigma\mathsf{QSO(SO)} & := & \alpha \text{ is restricted to } \textbf{sum operators} \text{ (i.e. } s, +, \Sigma x., \Sigma X.) \end{array}$$

Or both φ and α:

$$\begin{array}{rcl} \mathsf{QFO(LFP)} & = & \alpha \text{ is restricted to } \textbf{first order operators} \\ & & \text{and } \varphi \text{ is restricted to } \textbf{LFP logic}. \end{array}$$

Outline

Quantitative second order logic

QSO vs counting complexity

Below and beyond

Capturing a counting complexity class with QSO

- Recall that a counting complexity $\mathcal{C} \subseteq \{f : \Sigma^* \to \mathbb{N}\}$.

Capturing a counting complexity class with QSO

- Recall that a counting complexity $\mathcal{C} \subseteq \{f : \Sigma^* \to \mathbb{N}\}$.
- Let $\mathrm{enc}(\mathfrak{A})$ be any reasonable encoding of $\mathfrak{A}$ into a string in Σ^*.

Capturing a counting complexity class with QSO

- Recall that a counting complexity $\mathcal{C} \subseteq \{f : \Sigma^* \to \mathbb{N}\}$.
- Let $\text{enc}(\mathfrak{A})$ be any reasonable encoding of $\mathfrak{A}$ into a string in Σ^*.

Definition

Let $\mathcal{F}$ be a fragment or extension of QSO and $\mathcal{C}$ a counting complexity class.

Capturing a counting complexity class with QSO

- Recall that a counting complexity $\mathcal{C} \subseteq \{f : \Sigma^* \to \mathbb{N}\}$.
- Let $\mathrm{enc}(\mathfrak{A})$ be any reasonable encoding of $\mathfrak{A}$ into a string in Σ^*.

Definition

Let $\mathcal{F}$ be a fragment or extension of QSO and $\mathcal{C}$ a counting complexity class. Then $\mathcal{F}$ **captures** $\mathcal{C}$ over ordered **R**-structures if:

1. for every $\alpha \in \mathcal{F}$, there exists $f \in \mathcal{C}$ such that $[\![\alpha]\!](\mathfrak{A}) = f(\mathrm{enc}(\mathcal{A}))$ for every $\mathfrak{A} \in \textsc{Struct}[\mathbf{R}]$.

Capturing a counting complexity class with QSO

- Recall that a counting complexity $\mathcal{C} \subseteq \{f : \Sigma^* \to \mathbb{N}\}$.
- Let $\mathrm{enc}(\mathfrak{A})$ be any reasonable encoding of $\mathfrak{A}$ into a string in Σ^*.

Definition

Let $\mathcal{F}$ be a fragment or extension of QSO and $\mathcal{C}$ a counting complexity class. Then $\mathcal{F}$ **captures** $\mathcal{C}$ over ordered **R**-structures if:

1. for every $\alpha \in \mathcal{F}$, there exists $f \in \mathcal{C}$ such that $[\![\alpha]\!](\mathfrak{A}) = f(\mathrm{enc}(\mathcal{A}))$ for every $\mathfrak{A} \in \textsc{Struct}[\mathbf{R}]$.
2. for every $f \in \mathcal{C}$, there exists $\alpha \in \mathcal{F}$ such that $f(\mathrm{enc}(\mathcal{A})) = [\![\alpha]\!](\mathfrak{A})$ for every $\mathfrak{A} \in \textsc{Struct}[\mathbf{R}]$.

Capturing a counting complexity class with QSO

- Recall that a counting complexity $\mathcal{C} \subseteq \{f : \Sigma^* \to \mathbb{N}\}$.
- Let enc($\mathfrak{A}$) be any reasonable encoding of $\mathfrak{A}$ into a string in Σ^*.

Definition

Let $\mathcal{F}$ be a fragment or extension of QSO and $\mathcal{C}$ a counting complexity class. Then $\mathcal{F}$ **captures** $\mathcal{C}$ over ordered **R**-structures if:

1. for every $\alpha \in \mathcal{F}$, there exists $f \in \mathcal{C}$ such that $[\![\alpha]\!](\mathfrak{A}) = f(\text{enc}(\mathcal{A}))$ for every $\mathfrak{A} \in \textsc{Struct}[\mathbf{R}]$.
2. for every $f \in \mathcal{C}$, there exists $\alpha \in \mathcal{F}$ such that $f(\text{enc}(\mathcal{A})) = [\![\alpha]\!](\mathfrak{A})$ for every $\mathfrak{A} \in \textsc{Struct}[\mathbf{R}]$.

$\mathcal{F}$ **captures** $\mathcal{C}$ over ordered structures if $\mathcal{F}$ captures $\mathcal{C}$ over ordered **R**-structures **for every signature R**.

What counting classes can be captured by QSO?

Counting complexity classes

$$\left\{\begin{array}{c} \#\mathrm{P} \\ \textsc{SpanP} \\ \mathrm{FP} \\ \#\mathrm{L} \\ \#\mathrm{PSPACE} \\ \vdots \end{array}\right.$$

What counting classes can be captured by QSO?

Counting complexity classes $\left\{ \begin{array}{c} \#\mathrm{P} \\ \textsc{SpanP} \\ \mathrm{FP} \\ \#\mathrm{L} \\ \#\mathrm{PSPACE} \\ \vdots \end{array} \right.$

We show that most of these classes
can be captured by subfragments or extensions of QSO

How to capture $\#\mathrm{P}$?

How to capture $\#\mathrm{P}$?

$f \in \#\mathrm{P}$ iff there exists an **NP machine** M such that $f(x) = \#\mathrm{accepts}_M(x)$ for all $x \in \Sigma^*$.

How to capture $\#\mathrm{P}$?

$f \in \#\mathrm{P}$ iff there exists an **NP machine** M such that $f(x) = \#\mathrm{accepts}_M(x)$ for all $x \in \Sigma^*$.

$\Sigma\mathrm{QSO}(\mathrm{FO})$:= α restricted to **sum operators** (i.e. $s, +, \Sigma x., \Sigma X.$) and φ restricted to **FO logic**.

How to capture $\#\mathrm{P}$?

$f \in \#\mathrm{P}$ iff there exists an **NP machine** M such that $f(x) = \#\mathrm{accepts}_M(x)$ for all $x \in \Sigma^*$.

$\Sigma\mathrm{QSO(FO)}$:= α restricted to **sum operators** (i.e. $s, +, \Sigma x., \Sigma X.$) and φ restricted to **FO logic**.

Theorem

$\Sigma\mathrm{QSO(FO)}$ captures $\#\mathrm{P}$ over ordered structures.

How to capture $\textsc{SpanP}$?

$$\#\mathrm{P} \quad \equiv \quad \Sigma\mathsf{QSO}(\mathsf{FO})$$

How to capture SpanP?

$$\#\mathrm{P} \quad \equiv \quad \Sigma\mathsf{QSO}(\mathsf{FO})$$

$f \in \textsc{SpanP}$ iff there exists an **NP machine** M with **output** such that $f(x) = \#\mathrm{outputs}_M(x)$ for all $x \in \Sigma^*$.

How to capture $\textsc{SpanP}$?

$$\#\mathrm{P} \quad \equiv \quad \Sigma\mathsf{QSO}(\mathsf{FO})$$

$f \in \textsc{SpanP}$ iff there exists an **NP machine** M with **output** such that $f(x) = \#\text{outputs}_M(x)$ for all $x \in \Sigma^*$.

$\Sigma\mathsf{QSO}(\exists\mathsf{SO}) \quad :=$ α restricted to **sum operators** (i.e. $s, +, \Sigma x., \Sigma X.$) and φ restricted to **existential SO logic**.

How to capture SpanP?

$$\#\mathrm{P} \quad \equiv \quad \Sigma\mathsf{QSO}(\mathsf{FO})$$

$f \in \textsc{SpanP}$ iff there exists an **NP machine** M with **output** such that $f(x) = \#\text{outputs}_M(x)$ for all $x \in \Sigma^*$.

$\Sigma\mathsf{QSO}(\exists\mathsf{SO})$:= α restricted to **sum operators** (i.e. $s, +, \Sigma x., \Sigma X.$) and φ restricted to **existential SO logic**.

Theorem

$\Sigma\mathsf{QSO}(\exists\mathsf{SO})$ captures $\textsc{SpanP}$ over ordered structures.

How to capture SpanP?

$$\#\mathrm{P} \quad \equiv \quad \Sigma\mathsf{QSO}(\mathsf{FO})$$

$f \in \textsc{SpanP}$ iff there exists an **NP machine** M with **output** such that $f(x) = \#\text{outputs}_M(x)$ for all $x \in \Sigma^*$.

$\Sigma\mathsf{QSO}(\exists\mathsf{SO})$:= α restricted to **sum operators** (i.e. $s, +, \Sigma x., \Sigma X.$) and φ restricted to **existential SO logic**.

Theorem

$\Sigma\mathsf{QSO}(\exists\mathsf{SO})$ captures SpanP over ordered structures.

$\#\mathrm{P}$ and SpanP were shown to be captured by a different framework of Saluja et al. and Compton et al.

How to capture FP?

$$
\begin{array}{rcl}
\#\mathrm{P} & \equiv & \Sigma\mathsf{QSO}(\mathsf{FO}) \\
\textsc{SpanP} & \equiv & \Sigma\mathsf{QSO}(\exists\mathsf{SO})
\end{array}
$$

How to capture FP?

$$\begin{array}{rcl} \#\mathrm{P} & \equiv & \Sigma\mathsf{QSO}(\mathsf{FO}) \\ \textsc{SpanP} & \equiv & \Sigma\mathsf{QSO}(\exists\mathsf{SO}) \end{array}$$

$f \in \mathrm{FP}$ iff there exists a **PTIME machine** M with output such that $f(x) = M(x)$ for all $x \in \Sigma^*$.

How to capture FP?

$$\begin{array}{rcl} \#\mathrm{P} & \equiv & \Sigma\mathsf{QSO}(\mathsf{FO}) \\ \textsc{SpanP} & \equiv & \Sigma\mathsf{QSO}(\exists\mathsf{SO}) \end{array}$$

$f \in \mathrm{FP}$ iff there exists a **PTIME machine** M with output such that $f(x) = M(x)$ for all $x \in \Sigma^*$.

QFO(LFP) $:=$ α restricted to **first order op.** (i.e. $+, \cdot, \Sigma x., \Pi x.$) and φ restricted to **LFP logic**.

How to capture FP?

$$\begin{array}{rcl} \#\mathrm{P} & \equiv & \Sigma\mathsf{QSO}(\mathsf{FO}) \\ \textsc{SpanP} & \equiv & \Sigma\mathsf{QSO}(\exists\mathsf{SO}) \end{array}$$

$f \in \mathrm{FP}$ iff there exists a **PTIME machine** M with output such that $f(x) = M(x)$ for all $x \in \Sigma^*$.

QFO(LFP) $:=$ α restricted to **first order op.** (i.e. $+, \cdot, \Sigma x., \Pi x.$) and φ restricted to **LFP logic**.

Theorem

QFO(LFP) captures FP over ordered structures.

How to capture FPSPACE?

$$
\begin{array}{ccc}
\#\mathrm{P} & \equiv & \Sigma\mathsf{QSO}(\mathsf{FO}) \\
\textsc{SpanP} & \equiv & \Sigma\mathsf{QSO}(\exists\mathsf{SO}) \\
\mathrm{FP} & \equiv & \mathsf{QFO}(\mathsf{LFP})
\end{array}
$$

How to capture $\mathrm{FPSPACE}$?

$$\begin{array}{ccc} \#\mathrm{P} & \equiv & \Sigma\mathsf{QSO}(\mathsf{FO}) \\ \textsc{SpanP} & \equiv & \Sigma\mathsf{QSO}(\exists\mathsf{SO}) \\ \mathrm{FP} & \equiv & \mathsf{QFO}(\mathsf{LFP}) \end{array}$$

$f \in \mathrm{FPSPACE}$ iff there exists a **PSPACE machine** M with output such that $f(x) = M(x)$ for all $x \in \Sigma^*$.

How to capture $\mathrm{FPSPACE}$?

$$\begin{array}{ccc} \#\mathrm{P} & \equiv & \Sigma\mathsf{QSO}(\mathsf{FO}) \\ \textsc{SpanP} & \equiv & \Sigma\mathsf{QSO}(\exists\mathsf{SO}) \\ \mathrm{FP} & \equiv & \mathsf{QFO}(\mathsf{LFP}) \end{array}$$

$f \in \mathrm{FPSPACE}$ iff there exists a **PSPACE machine** M with output such that $f(x) = M(x)$ for all $x \in \Sigma^*$.

$\mathsf{QSO}(\mathsf{PFP})$:= φ restricted to **PFP logic**.

How to capture $\mathrm{FPSPACE}$?

$$
\begin{array}{ccc}
\#\mathrm{P} & \equiv & \Sigma\mathsf{QSO}(\mathsf{FO}) \\
\textsc{SpanP} & \equiv & \Sigma\mathsf{QSO}(\exists\mathsf{SO}) \\
\mathrm{FP} & \equiv & \mathsf{QFO}(\mathsf{LFP})
\end{array}
$$

$f \in \mathrm{FPSPACE}$ iff there exists a **PSPACE machine** M with output such that $f(x) = M(x)$ for all $x \in \Sigma^*$.

QSO(PFP) := φ restricted to **PFP logic**.

Theorem

QSO(PFP) captures $\mathrm{FPSPACE}$ over ordered structures.

How to capture $\mathrm{FPSPACE}(\text{poly})$?

$$
\begin{array}{ccc}
\#\mathrm{P} & \equiv & \Sigma\mathsf{QSO}(\mathsf{FO}) \\
\textsc{SpanP} & \equiv & \Sigma\mathsf{QSO}(\exists\mathsf{SO}) \\
\mathrm{FP} & \equiv & \mathsf{QFO}(\mathsf{LFP}) \\
\mathrm{FPSPACE} & \equiv & \mathsf{QSO}(\mathsf{PFP})
\end{array}
$$

How to capture $\mathrm{FPSPACE}(\text{poly})$?

$$\begin{array}{lcl} \#\mathrm{P} & \equiv & \Sigma\mathsf{QSO}(\mathsf{FO}) \\ \textsc{SpanP} & \equiv & \Sigma\mathsf{QSO}(\exists\mathsf{SO}) \\ \mathrm{FP} & \equiv & \mathsf{QFO}(\mathsf{LFP}) \\ \mathrm{FPSPACE} & \equiv & \mathsf{QSO}(\mathsf{PFP}) \end{array}$$

$f \in \mathrm{FPSPACE}(\text{poly})$ iff there exists a **PSPACE machine** M with output of polynomial size such that $f(x) = M(x)$ for all $x \in \Sigma^*$.

How to capture $\mathrm{FPSPACE}(\mathsf{poly})$?

$$\begin{array}{lcl} \#\mathrm{P} & \equiv & \Sigma\mathsf{QSO}(\mathsf{FO}) \\ \textsc{SpanP} & \equiv & \Sigma\mathsf{QSO}(\exists\mathsf{SO}) \\ \mathrm{FP} & \equiv & \mathsf{QFO}(\mathsf{LFP}) \\ \mathrm{FPSPACE} & \equiv & \mathsf{QSO}(\mathsf{PFP}) \end{array}$$

$f \in \mathrm{FPSPACE}(\mathsf{poly})$ iff there exists a **PSPACE machine** M with output of polynomial size such that $f(x) = M(x)$ for all $x \in \Sigma^*$.

$\mathsf{QFO}(\mathsf{PFP})$:= α restricted to **first order op.** (i.e. $+, \cdot, \Sigma x., \Pi x.$) and φ restricted to **PFP logic**.

How to capture $\mathrm{FPSPACE}(\mathsf{poly})$?

$$\begin{array}{lcl} \#\mathrm{P} & \equiv & \Sigma\mathsf{QSO}(\mathsf{FO}) \\ \textsc{SpanP} & \equiv & \Sigma\mathsf{QSO}(\exists\mathsf{SO}) \\ \mathrm{FP} & \equiv & \mathsf{QFO}(\mathsf{LFP}) \\ \mathrm{FPSPACE} & \equiv & \mathsf{QSO}(\mathsf{PFP}) \end{array}$$

$f \in \mathrm{FPSPACE}(\mathsf{poly})$ iff there exists a **PSPACE machine** M with output of polynomial size such that $f(x) = M(x)$ for all $x \in \Sigma^*$.

QFO(PFP) := α restricted to **first order op.** (i.e. $+, \cdot, \Sigma x., \Pi x.$) and φ restricted to **PFP logic**.

Theorem

QFO(PFP) captures $\mathrm{FPSPACE}(\mathsf{poly})$ over ordered structures.

More classes?

$$\begin{array}{rcl}
\#\mathrm{P} & \equiv & \Sigma\mathsf{QSO}(\mathsf{FO}) \\
\textsc{SpanP} & \equiv & \Sigma\mathsf{QSO}(\exists\mathsf{SO}) \\
\mathrm{FP} & \equiv & \mathsf{QFO}(\mathsf{LFP}) \\
\mathrm{FPSPACE} & \equiv & \mathsf{QSO}(\mathsf{PFP}) \\
\mathrm{FPSPACE}(\mathsf{poly}) & \equiv & \mathsf{QFO}(\mathsf{PFP})
\end{array}$$

More classes?

$$\begin{array}{ccc}
\#\mathrm{P} & \equiv & \Sigma\mathsf{QSO}(\mathsf{FO}) \\
\textsc{SpanP} & \equiv & \Sigma\mathsf{QSO}(\exists\mathsf{SO}) \\
\mathrm{FP} & \equiv & \mathsf{QFO}(\mathsf{LFP}) \\
\mathrm{FPSPACE} & \equiv & \mathsf{QSO}(\mathsf{PFP}) \\
\mathrm{FPSPACE}(\mathsf{poly}) & \equiv & \mathsf{QFO}(\mathsf{PFP}) \\
\textsc{GapP} & \equiv & \Sigma\mathsf{QSO}_{\mathbb{Z}}(\mathsf{FO})
\end{array}$$

More classes?

$$
\begin{array}{rcl}
\#\mathrm{P} & \equiv & \Sigma\mathsf{QSO}(\mathsf{FO}) \\
\textsc{SpanP} & \equiv & \Sigma\mathsf{QSO}(\exists\mathsf{SO}) \\
\mathrm{FP} & \equiv & \mathsf{QFO}(\mathsf{LFP}) \\
\mathrm{FPSPACE} & \equiv & \mathsf{QSO}(\mathsf{PFP}) \\
\mathrm{FPSPACE}(\mathsf{poly}) & \equiv & \mathsf{QFO}(\mathsf{PFP}) \\
\textsc{GapP} & \equiv & \Sigma\mathsf{QSO}_{\mathbb{Z}}(\mathsf{FO}) \\
\textsc{MaxP} & \equiv & \mathsf{MaxQSO}(\mathsf{FO}) \\
\textsc{MinP} & \equiv & \mathsf{MinQSO}(\mathsf{FO})
\end{array}
$$

Outline

Use QSO to understand classes **below** $\#\mathrm{P}$

$$\#\mathrm{P} \quad \equiv \quad \Sigma\mathsf{QSO}(\mathsf{FO})$$

Use QSO to understand classes **below** $\#\mathrm{P}$

$$\#\mathrm{P} \quad \equiv \quad \Sigma\mathsf{QSO}(\mathsf{FO})$$

We consider subfragments below FO:

$$\begin{aligned}
\Sigma_0 &= \{\ \theta \in \mathsf{FO} \mid \theta \text{ has no first-order quantifiers}\ \} \\
\Sigma_1 &= \{\ \varphi \in \mathsf{FO} \mid \varphi = \exists \bar{x}.\, \theta(\bar{x}) \ \wedge\ \theta \in \Sigma_0\ \} \\
\Pi_1 &= \{\ \varphi \in \mathsf{FO} \mid \varphi = \forall \bar{x}.\, \theta(\bar{x}) \ \wedge\ \theta \in \Sigma_0\ \} \\
\Sigma_2 &= \{\ \varphi \in \mathsf{FO} \mid \varphi = \exists \bar{x}.\, \forall \bar{y}.\, \theta(\bar{x}, \bar{y}) \ \wedge\ \theta \in \Sigma_0\ \} \\
\Pi_2 &= \{\ \varphi \in \mathsf{FO} \mid \varphi = \forall \bar{x}.\, \exists \bar{y}.\, \theta(\bar{x}, \bar{y}) \ \wedge\ \theta \in \Sigma_0\ \}
\end{aligned}$$

Use QSO to understand classes **below** $\#\mathrm{P}$

$$\#\mathrm{P} \quad \equiv \quad \Sigma\mathsf{QSO}(\mathsf{FO})$$

Saluja et. al. counting classes below $\#\mathrm{P}$

$$\#\Sigma_0 \subsetneq \#\Sigma_1 \subsetneq \#\Pi_1 \subsetneq \#\Sigma_2 \subsetneq \#\Pi_2 = \#\mathsf{FO} \equiv \#\mathrm{P}$$

Use QSO to understand classes **below** $\#\mathrm{P}$

$$\#\mathrm{P} \quad \equiv \quad \Sigma\mathsf{QSO}(\mathsf{FO})$$

Saluja et. al. counting classes below $\#\mathrm{P}$

$$\#\Sigma_0 \subsetneq \#\Sigma_1 \subsetneq \#\Pi_1 \subsetneq \#\Sigma_2 \subsetneq \#\Pi_2 = \#\mathsf{FO} \equiv \#\mathrm{P}$$

Theorem (ΣQSO-hierarchy)

$$\begin{tikzcd}[column sep=small]
 & \#\Sigma_1 \\
\#\Sigma_0 \arrow[ur, phantom, sloped, "\subsetneq"] & \\
 & \Sigma\mathsf{QSO}(\Sigma_0) \arrow[ul, phantom, sloped, "\subsetneq"]
\end{tikzcd}$$

Use QSO to understand classes **below** $\#\mathrm{P}$

$$\#\mathrm{P} \quad \equiv \quad \Sigma\mathsf{QSO}(\mathsf{FO})$$

Saluja et. al. counting classes below $\#\mathrm{P}$

$$\#\Sigma_0 \subsetneq \#\Sigma_1 \subsetneq \#\Pi_1 \subsetneq \#\Sigma_2 \subsetneq \#\Pi_2 = \#\mathsf{FO} \equiv \#\mathrm{P}$$

Theorem (ΣQSO-hierarchy)

$$\begin{tikzcd}
 & \#\Sigma_1 & \\
\#\Sigma_0 \arrow[ur, phantom, sloped, "\subsetneq"] & & \Sigma\mathsf{QSO}(\Sigma_1) \arrow[ul, phantom, sloped, "\subsetneq"] \\
 & \Sigma\mathsf{QSO}(\Sigma_0) \arrow[ul, phantom, sloped, "\subsetneq"] \arrow[ur, phantom, sloped, "\subsetneq"] &
\end{tikzcd}$$

Use QSO to understand classes **below** $\#\mathrm{P}$

$$\#\mathrm{P} \quad \equiv \quad \Sigma\mathsf{QSO}(\mathsf{FO})$$

Saluja et. al. counting classes below $\#\mathrm{P}$

$$\#\Sigma_0 \subsetneq \#\Sigma_1 \subsetneq \#\Pi_1 \subsetneq \#\Sigma_2 \subsetneq \#\Pi_2 = \#\mathsf{FO} \equiv \#\mathrm{P}$$

Theorem (ΣQSO-hierarchy)

$$\begin{tikzcd}
 & \#\Sigma_1 \arrow[dr, phantom, sloped, "\subsetneq"] & & & \\
\#\Sigma_0 \arrow[ur, phantom, sloped, "\subsetneq"] \arrow[dr, phantom, sloped, "\subsetneq"] & & \Sigma\mathsf{QSO}(\Sigma_1) \arrow[r, phantom, "\subsetneq"] & \#\Pi_1 \arrow[r, phantom, "="] & \Sigma\mathsf{QSO}(\Pi_1) \\
 & \Sigma\mathsf{QSO}(\Sigma_0) \arrow[ur, phantom, sloped, "\subsetneq"] & & &
\end{tikzcd}$$

Use QSO to understand classes **below** $\#\mathrm{P}$

$$\#\mathrm{P} \quad \equiv \quad \Sigma\mathrm{QSO}(\mathrm{FO})$$

Saluja et. al. counting classes below $\#\mathrm{P}$

$$\#\Sigma_0 \subsetneq \#\Sigma_1 \subsetneq \#\Pi_1 \subsetneq \#\Sigma_2 \subsetneq \#\Pi_2 = \#\mathrm{FO} \equiv \#\mathrm{P}$$

Theorem (ΣQSO-hierarchy)

$$\begin{tikzcd}
 & \#\Sigma_1 \arrow[dr, phantom, sloped, "\subsetneq"] & & & & & & & \\
\#\Sigma_0 \arrow[ur, phantom, sloped, "\subsetneq"] \arrow[dr, phantom, sloped, "\subsetneq"] & & \Sigma\mathrm{QSO}(\Sigma_1) \arrow[r, phantom, "\subsetneq"] & \#\Pi_1 \arrow[r, phantom, "="] & \Sigma\mathrm{QSO}(\Pi_1) \arrow[r, phantom, "\subsetneq"] & \#\Sigma_2 \arrow[ddllll, bend left] \\
 & \Sigma\mathrm{QSO}(\Sigma_0) \arrow[ur, phantom, sloped, "\subsetneq"] & & & & \\
 & \#\Sigma_2 = \Sigma\mathrm{QSO}(\Sigma_2) & & & &
\end{tikzcd}$$

Use QSO to understand classes **below** $\#\mathrm{P}$

$$\#\mathrm{P} \quad \equiv \quad \Sigma\mathsf{QSO}(\mathsf{FO})$$

Saluja et. al. counting classes below $\#\mathrm{P}$

$$\#\Sigma_0 \subsetneq \#\Sigma_1 \subsetneq \#\Pi_1 \subsetneq \#\Sigma_2 \subsetneq \#\Pi_2 = \#\mathsf{FO} \equiv \#\mathrm{P}$$

Theorem (ΣQSO-hierarchy)

$$\begin{tikzcd}
 & \#\Sigma_1 \arrow[dr, phantom, "\subsetneq"] & & & & & & & \\
\#\Sigma_0 \arrow[ur, phantom, "\subsetneq"] \arrow[dr, phantom, "\subsetneq"] & & \Sigma\mathsf{QSO}(\Sigma_1) \arrow[r, phantom, "\subsetneq"] & \#\Pi_1 \arrow[r, phantom, "="] & \Sigma\mathsf{QSO}(\Pi_1) \arrow[r, phantom, "\subsetneq"] & \#\Sigma_2 \\
 & \Sigma\mathsf{QSO}(\Sigma_0) \arrow[ur, phantom, "\subsetneq"] & & & & &
\end{tikzcd}$$

$$\#\Sigma_2 = \Sigma\mathsf{QSO}(\Sigma_2) \subsetneq \#\Pi_2 = \Sigma\mathsf{QSO}(\Pi_2) \equiv \#\mathrm{P}$$

The class $\Sigma\mathsf{QSO}(\Sigma_1[\mathsf{FO}])$

The class $\Sigma\mathrm{QSO}(\Sigma_1[\mathrm{FO}])$

Consider the following fragment of FO:

The class $\Sigma\mathrm{QSO}(\Sigma_1[\mathrm{FO}])$

Consider the following fragment of FO:

$$\Sigma_1[\mathrm{FO}] \quad = \quad \{\ \varphi \in \mathrm{FO} \ \mid\ \varphi = \exists \bar{x}.\, \theta(\bar{x}) \text{ and } \theta \text{ can contain atomic formulae of the form } u = v\,,\ X(\bar{u}) \textbf{ and } \boldsymbol{\varphi(\bar{u}) \in \mathrm{FO}}\ \}$$

The class $\Sigma\mathrm{QSO}(\Sigma_1[\mathrm{FO}])$

Consider the following fragment of FO:

$$\Sigma_1[\mathrm{FO}] \quad = \quad \{\ \varphi \in \mathrm{FO} \mid \varphi = \exists \bar{x}.\, \theta(\bar{x}) \text{ and } \theta \text{ can contain atomic formulae of the form } u = v\,,\ X(\bar{u}) \textbf{ and } \boldsymbol{\varphi(\bar{u}) \in \mathrm{FO}}\ \}$$

Theorem (good decision and closure properties)

The class $\Sigma\mathrm{QSO}(\Sigma_1[\mathrm{FO}])$ is closed under sum, multiplication and **subtraction by one**.

The class $\Sigma\mathrm{QSO}(\Sigma_1[\mathrm{FO}])$

Consider the following fragment of FO:

$$\Sigma_1[\mathrm{FO}] \quad = \quad \{\ \varphi \in \mathrm{FO} \ \mid\ \varphi = \exists \bar{x}.\, \theta(\bar{x}) \text{ and } \theta \text{ can contain atomic formulae of the form } u = v\,,\ X(\bar{u}) \textbf{ and } \boldsymbol{\varphi(\bar{u}) \in \mathrm{FO}}\ \}$$

Theorem (good decision and closure properties)

The class $\Sigma\mathrm{QSO}(\Sigma_1[\mathrm{FO}])$ is closed under sum, multiplication and **subtraction by one**. Moreover, $\Sigma\mathrm{QSO}(\Sigma_1[\mathrm{FO}]) \subseteq \mathrm{TotP}$

The class $\Sigma\text{QSO}(\Sigma_1[\text{FO}])$

Consider the following fragment of FO:

$$\Sigma_1[\text{FO}] \quad = \quad \{\ \varphi \in \text{FO} \ \mid \ \varphi = \exists \bar{x}.\, \theta(\bar{x}) \text{ and } \theta \text{ can contain atomic formulae of the form } u = v\,,\ X(\bar{u}) \textbf{ and } \boldsymbol{\varphi(\bar{u}) \in \text{FO}}\ \}$$

Theorem (good decision and closure properties)

The class $\Sigma\text{QSO}(\Sigma_1[\text{FO}])$ is closed under sum, multiplication and **subtraction by one**. Moreover, $\Sigma\text{QSO}(\Sigma_1[\text{FO}]) \subseteq \text{TotP}$ and every function in $\Sigma\text{QSO}(\Sigma_1[\text{FO}])$ has an FPRAS.

$\Sigma QSO(\Sigma_1[FO])$ is closed under subtraction by one

(proof sketch)

$\Sigma\mathsf{QSO}(\Sigma_1[\mathsf{FO}])$ is closed under subtraction by one

(proof sketch)

We focus on the case where $\alpha \in \Sigma\mathsf{QSO}(\Sigma_1[\mathsf{FO}])$ is of the form:

$$\alpha \quad = \quad \Sigma\bar{X}.\,\Sigma\bar{x}.\,\exists\bar{y}.\,\varphi(\bar{X},\bar{x},\bar{y})$$

$\Sigma\mathrm{QSO}(\Sigma_1[\mathrm{FO}])$ is closed under subtraction by one

(proof sketch)

We focus on the case where $\alpha \in \Sigma\mathrm{QSO}(\Sigma_1[\mathrm{FO}])$ is of the form:

$$\alpha \quad = \quad \Sigma\bar{X}.\, \Sigma\bar{x}.\, \exists\bar{y}.\, \varphi(\bar{X},\bar{x},\bar{y})$$

We construct a formula $\mathit{min}\text{-}\varphi^{\mathrm{FO}}(\bar{x})$ that identifies the **lexicographically minimal** assignment σ to $\bar{x}$ that satisfies $\exists\bar{X}.\, \exists\bar{y}.\, \varphi(\bar{X},\bar{x},\bar{y})$.

$\Sigma\mathrm{QSO}(\Sigma_1[\mathrm{FO}])$ is closed under subtraction by one

(proof sketch)

We focus on the case where $\alpha \in \Sigma\mathrm{QSO}(\Sigma_1[\mathrm{FO}])$ is of the form:

$$\alpha \quad = \quad \Sigma\bar{X}.\, \Sigma\bar{x}.\, \exists\bar{y}.\, \varphi(\bar{X},\bar{x},\bar{y})$$

We construct a formula $\mathit{min}\text{-}\varphi^{\mathsf{FO}}(\bar{x})$ that identifies the **lexicographically minimal** assignment σ to $\bar{x}$ that satisfies $\exists\bar{X}.\, \exists\bar{y}.\, \varphi(\bar{X},\bar{x},\bar{y})$.

Then we use $\mathit{min}\text{-}\varphi^{\mathsf{FO}}(\bar{x})$ to define a formula $\psi(\bar{X},\bar{x})$ that **filters out** the minimal assignment to $\bar{X}$ for that σ.

$\Sigma QSO(\Sigma_1[FO])$ is closed under subtraction by one

(proof sketch)

We focus on the case where $\alpha \in \Sigma QSO(\Sigma_1[FO])$ is of the form:

$$\alpha \quad = \quad \Sigma \bar{X}.\, \Sigma \bar{x}.\, \exists \bar{y}.\, \varphi(\bar{X}, \bar{x}, \bar{y})$$

We construct a formula $\mathit{min}\text{-}\varphi^{\mathsf{FO}}(\bar{x})$ that identifies the **lexicographically minimal** assignment σ to $\bar{x}$ that satisfies $\exists \bar{X}.\, \exists \bar{y}.\, \varphi(\bar{X}, \bar{x}, \bar{y})$.

Then we use $\mathit{min}\text{-}\varphi^{\mathsf{FO}}(\bar{x})$ to define a formula $\psi(\bar{X}, \bar{x})$ that **filters out** the minimal assignment to $\bar{X}$ for that σ.

Lastly, we define a formula that counts one assignment less for $(\bar{X}, \bar{x})$:

$$\alpha' \quad = \quad \Sigma \bar{X}.\, \Sigma \bar{x}.\, \exists \bar{y}.\, \varphi(\bar{X}, \bar{x}, \bar{y}) \wedge \psi(\bar{X}, \bar{x})$$

$\Sigma\mathsf{QSO}(\Sigma_1[\mathsf{FO}])$ is closed under subtraction by one

(proof sketch)

We focus on the case where $\alpha \in \Sigma\mathsf{QSO}(\Sigma_1[\mathsf{FO}])$ is of the form:

$$\alpha \quad = \quad \Sigma\bar{X}.\,\Sigma\bar{x}.\,\exists\bar{y}.\,\varphi(\bar{X},\bar{x},\bar{y})$$

We construct a formula $\mathit{min}\text{-}\varphi^{\mathsf{FO}}(\bar{x})$ that identifies the **lexicographically minimal** assignment σ to $\bar{x}$ that satisfies $\exists\bar{X}.\,\exists\bar{y}.\,\varphi(\bar{X},\bar{x},\bar{y})$.

Then we use $\mathit{min}\text{-}\varphi^{\mathsf{FO}}(\bar{x})$ to define a formula $\psi(\bar{X},\bar{x})$ that **filters out** the minimal assignment to $\bar{X}$ for that σ.

Lastly, we define a formula that counts one assignment less for $(\bar{X},\bar{x})$:

$$\alpha' \quad = \quad \Sigma\bar{X}.\,\Sigma\bar{x}.\,\exists\bar{y}.\,\varphi(\bar{X},\bar{x},\bar{y}) \wedge \psi(\bar{X},\bar{x})$$

In this setting, the existence of a **small witness** (in this case σ) is essential to have closure by subtraction by one.

Extend QSO to capture complexity classes **beyond** QSO

Extend QSO to capture complexity classes **beyond** QSO

We extend QFO with **recursion**:

$$\text{RQFO} \quad = \quad \text{QFO with } \textbf{quantitative} \text{ recursion.}$$

Extend QSO to capture complexity classes **beyond** QSO

We extend QFO with **recursion**:

RQFO	$=$	QFO with **quantitative** recursion.
TQFO	$=$	QFO with **quantitative** transitive closure.

Extend QSO to capture complexity classes **beyond** QSO

We extend QFO with **recursion**:

$$\begin{aligned} \text{RQFO} &= \text{QFO with } \textbf{quantitative} \text{ recursion.} \\ \text{TQFO} &= \text{QFO with } \textbf{quantitative} \text{ transitive closure.} \end{aligned}$$

Theorem

1. RQFO(FO) captures FP over the class of ordered structures.

Extend QSO to capture complexity classes **beyond** QSO

We extend QFO with **recursion**:

$$\begin{aligned}\text{RQFO} &= \text{QFO with } \textbf{quantitative} \text{ recursion.}\\ \text{TQFO} &= \text{QFO with } \textbf{quantitative} \text{ transitive closure.}\end{aligned}$$

Theorem

1. RQFO(FO) captures FP over the class of ordered structures.
2. TQFO(FO) captures $\#\mathrm{L}$ over the class of ordered structures.

Conclusions and future work

"We believe that **quantitative logics** *are the right framework for Descriptive complexity of* **counting complexity classes***."*

Conclusions and future work

> *"We believe that* **quantitative logics** *are the right framework for Descriptive complexity of* **counting complexity classes**.*"*

Plenty of open problems here . . .

Conclusions and future work

"We believe that **quantitative logics** *are the right framework for Descriptive complexity of* **counting complexity classes**.*"*

Plenty of open problems here . . .

1. **Logical characterization** of classes like $\textsc{TotP}, \textsc{SpanL}, \ldots$

Conclusions and future work

"We believe that **quantitative logics** *are the right framework for Descriptive complexity of* **counting complexity classes**.*"*

Plenty of open problems here ...

1. **Logical characterization** of classes like $\textsc{TotP}, \textsc{SpanL}, \ldots$
2. **Compl. characterization** of subfragments like $\mathsf{QSO}(\mathsf{FO}), \mathsf{QFO}(\mathsf{FO}), \ldots$

Conclusions and future work

"We believe that **quantitative logics** *are the right framework for Descriptive complexity of* **counting complexity classes**.*"*

Plenty of open problems here ...

1. **Logical characterization** of classes like $\textsc{TotP}, \textsc{SpanL}, \ldots$
2. **Compl. characterization** of subfragments like $\mathsf{QSO}(\mathsf{FO}), \mathsf{QFO}(\mathsf{FO}), \ldots$
3. Use quantitative logic to find complexity **classes with good properties**.

Conclusions and future work

"We believe that **quantitative logics** *are the right framework for Descriptive complexity of* **counting complexity classes**.*"*

Plenty of open problems here ...

1. **Logical characterization** of classes like $\textsc{TotP}, \textsc{SpanL}, \ldots$
2. **Compl. characterization** of subfragments like $\mathsf{QSO}(\mathsf{FO}), \mathsf{QFO}(\mathsf{FO}), \ldots$
3. Use quantitative logic to find complexity **classes with good properties**.
4. Understand the **expressiveness** of QSO and their subfragments.

Conclusions and future work

"We believe that **quantitative logics** *are the right framework for Descriptive complexity of* **counting complexity classes**.*"*

Plenty of open problems here ...

1. **Logical characterization** of classes like $\textsc{TotP}, \textsc{SpanL}, \ldots$
2. **Compl. characterization** of subfragments like $\mathsf{QSO}(\mathsf{FO}), \mathsf{QFO}(\mathsf{FO}), \ldots$
3. Use quantitative logic to find complexity **classes with good properties**.
4. Understand the **expressiveness** of QSO and their subfragments.

Thanks! Questions?