Positive first-order logic on words and graphs

Denis Kuperberg

CNRS, LIP, ENS Lyon, Plume Team

Chocola, 10 March 2022
First-Order Logic (FO)

**Signature**: Predicate symbols \((P_1, \ldots, P_n)\) with arities \(k_1, \ldots, k_n\).

**Syntax** of FO:

\[
\phi, \psi := P_i(x_1, \ldots, x_{k_i}) \mid \phi \lor \psi \mid \phi \land \psi \mid \neg \phi \mid \exists x.\phi \mid \forall x.\phi
\]

**Example**: For directed graphs, signature = one binary predicate \(E\).

**Graph class**: Cliques

**Formula**: \(\phi = \forall x. \forall y. E(x, y)\)

\(\psi = \neg \exists x. \forall y. E(x, y)\)

**Example graph**

Model of \(\phi\)

Model of \(\psi\)

2/18
First-Order Logic (FO)

**Signature:** Predicate symbols \((P_1, \ldots, P_n)\) with arities \(k_1, \ldots, k_n\).

**Syntax** of FO:

\[
\varphi, \psi ::= P_i(x_1, \ldots, x_{k_i}) | \varphi \lor \psi | \varphi \land \psi | \neg \varphi | \exists x. \varphi | \forall x. \varphi
\]

**Semantics** of \(\varphi\):

Structure \((X, R_1, \ldots, R_n)\) is accepted or rejected.
First-Order Logic (FO)

**Signature**: Predicate symbols \((P_1, \ldots, P_n)\) with arities \(k_1, \ldots, k_n\).

**Syntax** of FO:

\[
\varphi, \psi := P_i(x_1, \ldots, x_{k_i}) \mid \varphi \lor \psi \mid \varphi \land \psi \mid \neg \varphi \mid \exists x. \varphi \mid \forall x. \varphi
\]

**Semantics** of \(\varphi\):
Structure \((X, R_1, \ldots, R_n)\) is accepted or rejected.

**Example**: For directed graphs, signature = one binary predicate \(E\).

<table>
<thead>
<tr>
<th>Graph class</th>
<th>Cliques</th>
<th>No node points to everyone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formula</td>
<td>(\varphi = \forall x. \forall y. E(x, y))</td>
<td>(\psi = \neg \exists x. \forall y. E(x, y))</td>
</tr>
</tbody>
</table>

**Example graph**

Model of \(\varphi\)

Model of \(\psi\)
Positive versus Monotone

**Goal**: Understand the role of negation in FO, any signature.
Positive versus Monotone

**Goal**: Understand the role of negation in FO, any signature.

**Positive formula**: no \( \neg \)

**Example**: graphs containing a triangle.

**Monotone class of structures**: closed under adding tuples to relations.

**Monotone formula**: defines a monotone class of structures.

**Fact**: \( \phi \) positive \( \Rightarrow \) \( \phi \) monotone.

What about the converse?

**Motivation**: Logics with fixed points. Fixed points can only be applied to monotone \( \phi \). Hard to recognize \( \Rightarrow \) replace by positive \( \phi \), syntactic condition.
Positive versus Monotone

**Goal:** Understand the role of negation in FO, any signature.

**Positive formula:** no →

**Monotone class of structures:** closed under adding tuples to relations.

For graph classes: monotone = closed under adding edges.

**Example:** graphs containing a triangle.
Positive versus Monotone

**Goal**: Understand the role of negation in FO, any signature.

**Positive formula**: no \( \neg \)

**Monotone class of structures**: closed under adding tuples to relations.  
For graph classes: monotone = closed under adding edges.  
**Example**: graphs containing a triangle.  
**Monotone formula**: defines a monotone class of structures.
Positive versus Monotone

Goal: Understand the role of negation in FO, any signature.

Positive formula: no \neg

Monotone class of structures: closed under adding tuples to relations.

For graph classes: monotone = closed under adding edges.

Example: graphs containing a triangle.

Monotone formula: defines a monotone class of structures.

Fact: \varphi positive \Rightarrow \varphi monotone.
Positive versus Monotone

**Goal**: Understand the role of negation in FO, any signature.

**Positive formula**: no ¬

**Monotone class of structures**: closed under adding tuples to relations.

For graph classes: monotone = closed under adding edges.

**Example**: graphs containing a triangle.

**Monotone formula**: defines a monotone class of structures.

**Fact**: \( \varphi \) positive \( \Rightarrow \) \( \varphi \) monotone.

What about the converse?
Positive versus Monotone

**Goal**: Understand the role of negation in FO, any signature.

**Positive formula**: no \( \neg \)

**Monotone class of structures**: closed under adding tuples to relations.

For graph classes: monotone = closed under adding edges.

**Example**: graphs containing a triangle.

**Monotone formula**: defines a monotone class of structures.

**Fact**: \( \varphi \) positive \( \Rightarrow \) \( \varphi \) monotone.

What about the converse?

**Motivation**: Logics with fixed points.

Fixed points can only be applied to monotone \( \varphi \).

Hard to recognize \( \rightarrow \) replace by positive \( \varphi \), syntactic condition.
Lyndon’s theorem

**Theorem (Lyndon 1959)**

If $\varphi$ is monotone then $\varphi$ is equivalent to a positive formula.

**On graph classes:** $\text{FO-definable} + \text{monotone} \Rightarrow \text{FO-definable without } \neg$. 

- Ajtai, Gurevich 1987: lattices, probas, number theory, complexity, topology, very hard
- Stolboushkin 1995: EF games on grid-like structures, involved
- This work: EF games on words, elementary

4/18
Lyndon’s theorem

Theorem (Lyndon 1959)

If $\varphi$ is monotone then $\varphi$ is equivalent to a positive formula.

On graph classes: $\text{FO-definable} + \text{monotone} \Rightarrow \text{FO-definable without} \neg$.

⚠️ Only true if we accept infinite structures.
Lyndon’s theorem

**Theorem (Lyndon 1959)**

If \( \varphi \) is **monotone** then \( \varphi \) is equivalent to a **positive** formula.

**On graph classes:** \( \text{FO-definable} + \text{monotone} \Rightarrow \text{FO-definable without } \neg \).

⚠️ Only true if we accept **infinite** structures.

What happens if we consider only **finite** structures?

This was open for 28 years...
Lyndon’s theorem

Theorem (Lyndon 1959)

If $\varphi$ is monotone then $\varphi$ is equivalent to a positive formula.

On graph classes: FO-definable + monotone $\Rightarrow$ FO-definable without $\neg$.

⚠️ Only true if we accept **infinite** structures.

What happens if we consider only **finite** structures?

This was open for 28 years...  

**Theorem**: Lyndon’s theorem fails on **finite** structures:

- [Ajtai, Gurevich 1987]
  lattices, probas, number theory, complexity, topology, very hard

- [Stolboushkin 1995]
  EF games on grid-like structures, involved
Lyndon’s theorem

**Theorem (Lyndon 1959)**

If \( \varphi \) is monotone then \( \varphi \) is equivalent to a positive formula.

On graph classes: FO-definable + monotone \( \Rightarrow \) FO-definable without \( \neg \).

⚠️ Only true if we accept infinite structures.

What happens if we consider only finite structures?

This was open for 28 years...

**Theorem:** Lyndon’s theorem fails on finite structures:

- [Ajtai, Gurevich 1987]
  - lattices, probas, number theory, complexity, topology, very hard

- [Stolboushkin 1995]
  - EF games on grid-like structures, involved

- [This work]
  - EF games on words, elementary
Our results

**Finite Model Theory:**

Lyndon’s theorem *fails* on

- Finite words
- Finite graphs
- Finite structures *(elementary proof)*, several versions:
  - one monotone predicate
  - some monotone predicates
  - all monotone predicates = closure under surjective morphisms.
Our results

**Finite Model Theory:**

Lyndon’s theorem fails on

- **Finite words**
- Finite graphs
- Finite structures (elementary proof), several versions:
  - one monotone predicate
  - some monotone predicates
  - all monotone predicates = closure under surjective morphisms.

**Regular Language Theory:**

<table>
<thead>
<tr>
<th>Monotone FO languages</th>
<th>≠</th>
<th>Positive FO languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algebraic characterization</td>
<td></td>
<td>Logical characterization</td>
</tr>
<tr>
<td>Decidable membership</td>
<td></td>
<td>Undecidable membership</td>
</tr>
</tbody>
</table>
**FO on words, the usual way**

Words on alphabet $A = \{a, b, \ldots\}$: signature $(\leq, a, b, \ldots)$

\[
\begin{array}{cccccc}
  a & b & a & a & b \\
  \bullet & \to & \bullet & \to & \bullet & \to \\
\end{array}
\]

- $x \leq y$ means position $x$ is before position $y$.
- $a(x)$ means position $x$ is labelled by the letter $a$
FO on words, the usual way

Words on alphabet $A = \{ a, b[,\ldots]\}$: signature $(\leq, a, b[,\ldots])$

$\subseteq a, b[\ldots]$]

- $a \quad b \quad a \quad a \quad b$

$\bullet \rightarrow \bullet \rightarrow \bullet \rightarrow \bullet \rightarrow \bullet$

- $x \leq y$ means position $x$ is before position $y$.
- $a(x)$ means position $x$ is labelled by the letter $a$

Examples of formulas:
- $\exists x. a(x)$: words containing $a$. Language $A^* a A^*$.
- $\exists x, y. (x \leq y \land a(x) \land b(y))$. Language $A^* a A^* b A^*$.
FO on words, the usual way

Words on alphabet $A = \{a, b, \ldots \}$: signature $(\leq, a, b, \ldots)$

$\begin{array}{ccccccc}
 & a & & b & & a & & a & & b \\
 \cdot & \rightarrow & \cdot & \rightarrow & \cdot & \rightarrow & \cdot & \rightarrow & \cdot
\end{array}$

- $x \leq y$ means position $x$ is before position $y$.
- $a(x)$ means position $x$ is labelled by the letter $a$

Examples of formulas:
- $\exists x. a(x)$: words containing $a$. Language $A^*aA^*$.
- $\exists x, y. (x \leq y \land a(x) \land b(y))$. Language $A^*aA^*bA^*$.

Theorem

First-order languages form a strict subclass of regular languages.

Example: $(aa)^*$ is not FO-definable. (Proof later)
Background: FO-definable languages

FO-definable languages are well-understood.
Background: FO-definable languages

FO-definable languages are well-understood.

Theorem (Schützenberger, McNaughton, Papert)

A language \( L \subseteq A^* \) is FO-definable iff it is definable by:
Star-free expression \( \Leftrightarrow \) LTL \( \Leftrightarrow \) counter-free automaton \( \Leftrightarrow \) ...
Background: FO-definable languages

FO-definable languages are well-understood.

**Theorem (Schützenberger, McNaughton, Papert)**

A language $L \subseteq A^*$ is FO-definable iff it is definable by:

- Star-free expression $\iff$ LTL $\iff$ counter-free automaton $\iff \ldots$

**Intuition:** FO languages are “Aperiodic”: cannot count modulo

$L$ aperiodic: There is $n \in \mathbb{N}$ such that $\forall u, v, w \in A^*$:

$$uv^n w \in L \iff uv^{n+1} w \in L.$$
Background: FO-definable languages

FO-definable languages are well-understood.

Theorem (Schützenberger, McNaughton, Papert)

A language $L \subseteq A^*$ is FO-definable iff it is definable by:
Star-free expression $\iff$ LTL $\iff$ counter-free automaton $\iff \ldots$

Intuition: FO languages are “Aperiodic”: cannot count modulo
L aperiodic: There is $n \in \mathbb{N}$ such that $\forall u, v, w \in A^*$:

$$uv^n w \in L \iff uv^{n+1} w \in L.$$ 

$\iff$ Counter-free automaton: No cycle of the form:
Background: FO-definable languages

FO-definable languages are well-understood.

Theorem (Schützenberger, McNaughton, Papert)

A language $L \subseteq A^*$ is FO-definable iff it is definable by:
Star-free expression $\iff$ LTL $\iff$ counter-free automaton $\iff$ ...

Intuition: FO languages are “Aperiodic”: cannot count modulo
$L$ aperiodic: There is $n \in \mathbb{N}$ such that $\forall u, v, w \in A^*$:

$$uv^nw \in L \iff uv^{n+1}w \in L.$$ 

$\iff$ Counter-free automaton: No cycle of the form:

Corollary: FO-definability is decidable for regular languages.
FO on words, the “unconstrained” way

For now, a word is a structure \((X, \leq, a, b, \ldots)\) where

- \(\leq\) is a total order
- \(a, b, \ldots\) form a partition of \(X\).
FO on words, the “unconstrained” way

For now, a word is a structure \((X, \leq, a, b, \ldots)\) where

- \(\leq\) is a total order
- \(a, b, \ldots\) form a partition of \(X\).

Let us drop the second constraint: \(a, b, \ldots\) independent.
FO on words, the “unconstrained” way

For now, a word is a structure \((X, \leq, a, b, \ldots)\) where

- \(\leq\) is a total order
- \(a, b, \ldots\) form a partition of \(X\).

Let us drop the second constraint: \(a, b, \ldots\) independent.

→ Words on alphabet \(\mathcal{P}\{{a, b, \ldots}\}\):

\[
\emptyset \quad \{b\} \quad \{a, b\} \quad \emptyset \quad \{b\}
\]

We will note \(\Sigma = \{a, b, \ldots\}\), and \(A = \mathcal{P}(\Sigma)\) the alphabet.

→ Useful e.g. in verification (LTL,\ldots):
  independent signals can be true or false simultaneously.
FO on words, the “unconstrained” way

For now, a word is a structure \((X, \leq, a, b, \ldots)\) where

- \(\leq\) is a total order
- \(a, b, \ldots\) form a partition of \(X\).

Let us drop the second constraint: \(a, b, \ldots\) independent.

\(\rightarrow\) Words on alphabet \(\mathcal{P}\left(\{a, b, \ldots\}\right)\):

\[
\begin{align*}
\emptyset & \quad \{b\} & \quad \{a, b\} & \quad \emptyset & \quad \{b\} \\
\bullet & \quad \rightarrow & \quad \bullet & \quad \rightarrow & \quad \bullet & \quad \rightarrow & \quad \bullet & \quad \rightarrow & \quad \bullet
\end{align*}
\]

We will note \(\Sigma = \{a, b, \ldots\}\), and \(A = \mathcal{P}(\Sigma)\) the alphabet.

- Useful e.g. in verification (LTL,\ldots): independent signals can be true or false simultaneously.
- FO languages on alphabet \(A\) are the same (Preds=\(\Sigma\) or \(A\)).
FO on words, the “unconstrained” way

For now, a word is a structure \((X, \leq, a, b, \ldots)\) where

- \(\leq\) is a total order
- \(a, b, \ldots\) form a partition of \(X\).

Let us drop the second constraint: \(a, b, \ldots\) independent.

→ Words on alphabet \(\mathcal{P}(\{a, b, \ldots\})\):

\[
\emptyset \quad \{b\} \quad \{a, b\} \quad \emptyset \quad \{b\}
\]

We will note \(\Sigma = \{a, b, \ldots\}\), and \(A = \mathcal{P}(\Sigma)\) the alphabet.

- Useful e.g. in verification (LTL, \ldots): independent signals can be true or false simultaneously.
- FO languages on alphabet \(A\) are the same (Preds=\(\Sigma\) or \(A\)).
- We no longer have \(\neg a(x) \equiv \bigvee_{\beta \neq a} \beta(x)\).
FO on words, the “unconstrained” way

For now, a word is a structure \((X, \leq, a, b, \ldots)\) where

- \(\leq\) is a total order
- \(a, b, \ldots\) form a partition of \(X\).

Let us drop the second constraint: \(a, b, \ldots\) independent.

\(\rightarrow\) Words on alphabet \(\mathcal{P}(\{a, b, \ldots\}):\)

- \(\emptyset\)
- \(\{b\}\)
- \(\{a, b\}\)
- \(\emptyset\)
- \(\{b\}\)

\(\bullet \rightarrow \bullet \rightarrow \bullet \rightarrow \bullet \rightarrow \bullet \rightarrow \bullet\)

We will note \(\Sigma = \{a, b, \ldots\}\), and \(A = \mathcal{P}(\Sigma)\) the alphabet.

- Useful e.g. in verification (LTL, \ldots):
  independent signals can be true or false simultaneously.

- FO languages on alphabet \(A\) are the same (Preds=\(\Sigma\) or \(A\)).

- We no longer have \(-a(x) \equiv \bigvee_\beta \not\equiv_a \beta(x)\).
  \(\rightarrow\) Negation necessary for full FO.
The **FO$^+$** logic: **positive formulas**

**FO$^+$ Logic:** $a$ ranges over $\Sigma$, no $\neg$

$\varphi, \psi := a(x) \mid x \leq y \mid x < y \mid \varphi \lor \psi \mid \varphi \land \psi \mid \exists x. \varphi \mid \forall x. \varphi$

Example:

On $\Sigma = \{a, b\}$:

$\exists x, y. (x \leq y \land a(x) \land b(y)) \Rightarrow (A^*\{a\}A^*\{b\}) \cup (A^*\{a, b\}A^*)$

Remark:

$\emptyset^*$ undefinable in FO$^+$ (cannot say "\$a\$").

More generally:

FO$^+$ can only define monotone languages:

$u \alpha v \in L$ and $\alpha \subseteq \beta \Rightarrow u \beta v \in L$

Motivation:

abstraction of many logics not closed under $\neg$.

Question [Colcombet]: FO $\&$ monotone $\Rightarrow$ FO$^+$
The \( \text{FO}^+ \) logic: positive formulas

**\( \text{FO}^+ \) Logic:** \( a \) ranges over \( \Sigma \), no \( \neg \)

\[
\varphi, \psi := a(x) \mid x \leq y \mid x < y \mid \varphi \lor \psi \mid \varphi \land \psi \mid \exists x. \varphi \mid \forall x. \varphi
\]

**Example:** On \( \Sigma = \{a, b\} \):

\[
\exists x, y. (x \leq y) \land a(x) \land b(y) \rightsquigarrow (A^*\{a\}A^*\{b\}A^*) \cup (A^*\{a, b\}A^*)
\]
The $\text{FO}^+$ logic: positive formulas

$\text{FO}^+$ Logic: $a$ ranges over $\Sigma$, no $\neg$

$\varphi, \psi := a(x) \mid x \leq y \mid x < y \mid \varphi \lor \psi \mid \varphi \land \psi \mid \exists x. \varphi \mid \forall x. \varphi$

Example: On $\Sigma = \{a, b\}$:

$\exists x, y. (x \leq y) \land a(x) \land b(y) \leadsto (A^*\{a\}A^*\{b\}A^*) \cup (A^*\{a, b\}A^*)$

Remark: $\emptyset^*$ undefinable in $\text{FO}^+$ (cannot say "$\neg a$").
The $\text{FO}^+$ logic: positive formulas

**FO$^+$ Logic:** $a$ ranges over $\Sigma$, no $\neg$

$$\varphi, \psi := a(x) \mid x \leq y \mid x < y \mid \varphi \lor \psi \mid \varphi \land \psi \mid \exists x. \varphi \mid \forall x. \varphi$$

**Example:** On $\Sigma = \{a, b\}$:

$$\exists x, y. (x \leq y) \land a(x) \land b(y) \leadsto (A^*\{a\}A^*\{b\}A^*) \cup (A^*\{a, b\}A^*)$$

**Remark:** $\emptyset^*$ undefinable in $\text{FO}^+$ (cannot say "$\neg a$").

More generally: $\text{FO}^+$ can only define monotone languages:

$$u\alpha v \in L \text{ and } \alpha \subseteq \beta \Rightarrow u\beta v \in L$$
The $\text{FO}^+$ logic: positive formulas

$\text{FO}^+$ Logic: $a$ ranges over $\Sigma$, no $\neg$

$$\varphi, \psi := a(x) \mid x \leq y \mid x < y \mid \varphi \lor \psi \mid \varphi \land \psi \mid \exists x. \varphi \mid \forall x. \varphi$$

**Example:** On $\Sigma = \{a, b\}$:

$$\exists x, y. (x \leq y) \land a(x) \land b(y) \leadsto (A^*\{a\}A^*\{b\}A^*) \cup (A^*\{a, b\}A^*)$$

**Remark:** $\emptyset^*$ undefinable in $\text{FO}^+$ (cannot say "$\neg a$").

More generally: $\text{FO}^+$ can only define monotone languages:

$$u\alpha v \in L \text{ and } \alpha \subseteq \beta \Rightarrow u\beta v \in L$$

**Motivation:** abstraction of many logics not closed under $\neg$. 
The $\text{FO}^+$ logic: positive formulas

$\text{FO}^+$ Logic: $a$ ranges over $\Sigma$, no $\neg$

$$\varphi, \psi := a(x) \mid x \leq y \mid x < y \mid \varphi \lor \psi \mid \varphi \land \psi \mid \exists x. \varphi \mid \forall x. \varphi$$

Example: On $\Sigma = \{a, b\}$:

$$\exists x, y. (x \leq y) \land a(x) \land b(y) \leadsto (A^*\{a\}A^*\{b\}A^*) \cup (A^*\{a, b\}A^*)$$

Remark: $\emptyset^*$ undefinable in $\text{FO}^+$ (cannot say "$\neg a$").

More generally: $\text{FO}^+$ can only define monotone languages:

$$u\alpha v \in L \text{ and } \alpha \subseteq \beta \Rightarrow u\beta v \in L$$

Motivation: abstraction of many logics not closed under $\neg$.

Question [Colcombet]: $\text{FO} \& \text{ monotone } \Rightarrow \text{FO}^+$
A counter-example language

Our first result

There is $L$ monotone, FO-definable but not $\text{FO}^+$-definable.
A counter-example language

Our first result

There is $L$ monotone, FO-definable but not FO$^+$-definable.

Alphabet $A = \{\emptyset, a, b, c, (a)_b, (b)_c, (c)_a, (a)_b (c)_a\}$. Let $a^{\uparrow} = \{a, (a)_b, (c)_a\}$. 
A counter-example language

Our first result

There is \( L \) monotone, FO-definable but not FO\(^+\)-definable.

Alphabet \( A = \{ \emptyset, a, b, c, (a)_b, (b)_c, (c)_a, (a)_b (b)_c \} \). Let \( a^\uparrow = \{ a, (a)_b, (c)_a \} \).

Language \( L = (a^\uparrow b^\uparrow c^\uparrow)^* \bigcup A^* \left( \frac{a}{b} \right) A^* \).
A counter-example language

Our first result

There is $L$ monotone, FO-definable but not FO$^+$-definable.

Alphabet $A = \{\emptyset, a, b, c, (a_b), (b_c), (c_a), (a_b_c)\}$. Let $a^\uparrow = \{a, (a_b), (c_a)\}$.

Language $L = (a^\uparrow b^\uparrow c^\uparrow)^* \cup A^* \left(\frac{a}{b_c}\right) A^*$. Monotone
A counter-example language

Our first result

There is $L$ monotone, FO-definable but not $\text{FO}^{+}$-definable.

Alphabet $A = \{\emptyset, a, b, c, (a_b), (b_c), (c_a), (a_b_c)\}$. Let $a^\uparrow = \{a, (a_b), (c_a)\}$.

Language $L = (a^\uparrow b^\uparrow c^\uparrow)^* \cup A^* \left(\frac{a}{b_c}\right) A^*$. Monotone

Lemma: $L$ is FO-definable.

Proof: $L$ is counter-free. (no cycle labelled $\nu \geq 2$)
A counter-example language

Our first result

There is $L$ monotone, FO-definable but not $\text{FO}^+$-definable.

Alphabet $A = \{\emptyset, a, b, c, (a)_b, (b)_c, (c)_a, (a_b)_c\}$. Let $a^\uparrow = \{a, (a)_b, (c)_a\}$.

Language $L = (a^\uparrow b^\uparrow c^\uparrow)^* \cup A^* (a_b)_c A^*$. Monotone

Lemma: $L$ is FO-definable.

Proof: is counter-free. (no cycle labelled $\nu \geq 2$)

To prove $L$ is not $\text{FO}^+$-definable: Ehrenfeucht-Fraïssé games.
Ehrenfeucht-Fraïssé games for FO

Definition (EF games)
Played on two words $u, v$. At each round $i$:

- **Spoiler** places token $i$ in $u$ or $v$.
- **Duplicator** must answer token $i$ in the other word such that
  - the letter on token $i$ is the same in $u$ and $v$.
  - the tokens are in the same order in $u$ and $v$.

Let us note $u \equiv^n v$ if Duplicator can survive $n$ rounds on $u, v$.

Theorem (Ehrenfeucht, Fraïssé, 1950-1961)
$L$ not FO-definable $\iff$ For all $n$, there are $u \in L, v \not\in L$ s.t. $u \equiv^n v$.

Example
Proving $(aa)^\ast$ is not FO-definable:
$u = a^{2k} \in (aa)^\ast$: 

```
a a a a a a a a a a
```

$v = a^{2k-1} \not\in (aa)^\ast$: 

```
a a a a a a a a
```
Ehrenfeucht-Fraïssé games for FO

Definition (EF games)
Played on two words \( u, v \). At each round \( i \):

- **Spoiler** places token \( i \) in \( u \) or \( v \).
- **Duplicator** must answer token \( i \) in the other word such that
  - the letter on token \( i \) is the same in \( u \) and \( v \).
  - the tokens are in the same order in \( u \) and \( v \).

Let us note \( u \equiv_n v \) if **Duplicator** can survive \( n \) rounds on \( u, v \).
Ehrenfeucht-Fraïssé games for FO

Definition (EF games)
Played on two words $u, v$. At each round $i$:

- **Spoiler** places token $i$ in $u$ or $v$.
- **Duplicator** must answer token $i$ in the other word such that
  - the letter on token $i$ is the same in $u$ and $v$.
  - the tokens are in the same order in $u$ and $v$.

Let us note $u \equiv_n v$ if Duplicator can survive $n$ rounds on $u, v$.

Theorem (Ehrenfeucht,Fraïssé, 1950-1961)
$L$ not FO-definable $\iff$ For all $n$, there are $u \in L$, $v \notin L$ s.t. $u \equiv_n v$. 
Ehrenfeucht-Fraïssé games for FO

Definition (EF games)
Played on two words $u, v$. At each round $i$:

- **Spoiler** places token $i$ in $u$ or $v$.
- **Duplicator** must answer token $i$ in the other word such that
  - the letter on token $i$ is the same in $u$ and $v$.
  - the tokens are in the same order in $u$ and $v$.

Let us note $u \equiv_n v$ if Duplicator can survive $n$ rounds on $u, v$.

Theorem (Ehrenfeucht,Fraïssé, 1950-1961)
$L$ not FO-definable $\iff$ For all $n$, there are $u \in L$, $v \notin L$ s.t. $u \equiv_n v$.

Example
Proving $(aa)^*$ is not FO-definable:

$u = a^{2k} \in (aa)^* : \ a\ a\ a\ a\ a\ a\ a\ a\ a$
$v = a^{2k-1} \notin (aa)^* : \ a\ a\ a\ a\ a\ a\ a\ a\ a$
Proving $\text{FO}^+$-undefinability

Definition ($\text{EF}^+$ games)

New rule:
Letters in $u$ just have to be included in corresponding ones in $v$.

We write $u \preceq_n v$ if Duplicator can survive $n$ rounds.

Theorem (Correctness of $\text{EF}^+$ games)

$L$ not $\text{FO}^+$-definable $\iff \forall n, \exists u \in L, v \not\in L : u \preceq_n v$.

[Stolboushkin 1995 + this work]

Application: Proving $L$ is not $\text{FO}^+$-definable

$u \in L$: 

$$a \ b \ c \ a \ b \ c \ a \ b \ c$$

$v \not\in L$: 

$$(a \ b) (b \ c) (c \ a) (a \ b) (b \ c) (c \ a) (a \ b) (b \ c)$$
Proving $\text{FO}^+$-undefinability

**Definition ($\text{EF}^+$ games)**

New rule:
Letters in $u$ just have to be included in corresponding ones in $v$.

We write $u \preceq_n v$ if Duplicator can survive $n$ rounds.

**Theorem (Correctness of $\text{EF}^+$ games)**

$L$ not $\text{FO}^+$-definable $\iff \forall n, \text{there are } u \in L, v \notin L \text{ s.t. } u \preceq_n v$.

[Stolboushkin 1995 + this work]
Proving $\text{FO}^+$-undefinability

**Definition ($\text{EF}^+$ games)**

New rule:
Letters in $u$ just have to be included in corresponding ones in $v$.

We write $u \preceq_n v$ if Duplicator can survive $n$ rounds.

**Theorem (Correctness of $\text{EF}^+$ games)**

$L$ not $\text{FO}^+$-definable $\iff \forall n$, there are $u \in L$, $v \notin L$ s.t. $u \preceq_n v$.

[Stolboushkin 1995+this work]

**Application: Proving $L$ is not $\text{FO}^+$-definable**

\[
\begin{align*}
  u \in L : & \quad a \ b \ c \ a \ b \ c \ a \ b \ c \\
  v \notin L : & \quad (a)_b \ (b)_c \ (c)_a \ (a)_b \ (b)_c \ (c)_a \ (a)_b \ (b)_c
\end{align*}
\]
From finite words to finite structures.

**Goal:** Lift $L$ to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.
From finite words to finite structures.

**Goal:** Lift \( L \) to finite structures.
For now: signature \((\leq, a, b, c)\) assuming \( \leq \) is a total order.

**Several monotone predicates**

Axiomatize in FO that \( \leq \) is a total order.
From finite words to finite structures.

**Goal:** Lift \( L \) to finite structures.
For now: signature \((\leq, a, b, c)\) assuming \( \leq \) is a total order.

**Several monotone predicates**

Axiomatize in FO that \( \leq \) is a total order.
\( a, b, c \) are monotone but not \( \leq \).
From finite words to finite structures.

**Goal:** Lift $L$ to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

**Several monotone predicates**
Axiomatize in FO that $\leq$ is a total order.
$a, b, c$ are monotone but not $\leq$.

**One monotone predicate**
Alphabet encoded by one binary predicate $A$.

\[
\begin{align*}
    a(x) &\equiv A(0, x) & b(x) &\equiv A(1, x) & c(x) &\equiv A(2, x)
\end{align*}
\]
From finite words to finite structures.

**Goal:** Lift $L$ to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

**Several monotone predicates**
Axiomatize in FO that $\leq$ is a total order.
$a, b, c$ are monotone but not $\leq$.

**One monotone predicate**
Alphabet encoded by one binary predicate $A$.

$$ a(x) \equiv A(0, x) \quad b(x) \equiv A(1, x) \quad c(x) \equiv A(2, x) $$

$A$ is monotone but not $\leq$. 
From finite words to finite structures.

**Goal:** Lift $L$ to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

**Several monotone predicates**
Axiomatize in FO that $\leq$ is a total order.
$a, b, c$ are **monotone** but not $\leq$.

**One monotone predicate**
Alphabet encoded by one binary predicate $A$.

\[
a(x) \equiv A(0, x) \quad b(x) \equiv A(1, x) \quad c(x) \equiv A(2, x)
\]

$A$ is **monotone** but not $\leq$.

**All monotone predicates = closure under surjective morphisms**

**Problem:** We cannot say that $\leq$ is total in a monotone way.
From finite words to finite structures.

**Goal:** Lift \( L \) to finite structures.  
For now: signature \((\leq, a, b, c)\) assuming \(\leq\) is a total order.

**Several monotone predicates**

Axiomatize in FO that \(\leq\) is a total order.  
\(a, b, c\) are monotone but not \(\leq\).

**One monotone predicate**

Alphabet encoded by one binary predicate \(A\).

\[
a(x) \equiv A(0, x) \quad b(x) \equiv A(1, x) \quad c(x) \equiv A(2, x)
\]

\(A\) is monotone but not \(\leq\).

**All monotone predicates = closure under surjective morphisms**

**Problem:** We cannot say that \(\leq\) is total in a monotone way.  
**Solution:** Introduce a predicate \(\not\leq\).
From finite words to finite structures.

**Goal:** Lift $L$ to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

**Several monotone predicates**
Axiomatize in FO that $\leq$ is a total order.
$a, b, c$ are monotone but not $\leq$.

**One monotone predicate**
Alphabet encoded by one binary predicate $A$.

$$
a(x) \equiv A(0, x) \quad b(x) \equiv A(1, x) \quad c(x) \equiv A(2, x)
$$

$A$ is monotone but not $\leq$.

**All monotone predicates = closure under surjective morphisms**

**Problem:** We cannot say that $\leq$ is total in a monotone way.

**Solution:** Introduce a predicate $\not\leq$.

- Require $\forall x, y. (x \leq y) \lor (x \not\leq y)$
- If $\exists x, y. (x \leq y) \land (x \not\leq y)$ → accept
- Axiomatize that $\leq$ is total assuming $\not\leq$ is its complement.
From finite words to finite structures.

**Goal:** Lift $L$ to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

**Several monotone predicates**
Axiomatize in FO that $\leq$ is a total order.
$a, b, c$ are monotone but not $\leq$.

**One monotone predicate**
Alphabet encoded by one binary predicate $A$.

$$
a(x) \equiv A(0, x) \quad b(x) \equiv A(1, x) \quad c(x) \equiv A(2, x)
$$

$A$ is monotone but not $\leq$.

**All monotone predicates $=$ closure under surjective morphisms**

**Problem:** We cannot say that $\leq$ is total in a monotone way.

**Solution:** Introduce a predicate $\not\leq$.

- Require $\forall x, y. (x \leq y) \lor (x \not\leq y)$
- If $\exists x, y. (x \leq y) \land (x \not\leq y) \rightarrow$ accept
- Axiomatize that $\leq$ is total assuming $\not\leq$ is its complement.

$a, b, c, \leq, \not\leq$ are monotone.
From finite words to finite graphs

Encode words into (directed) graphs, here $ab (a b) c$:

$x a x b x c \rightarrow \text{formula } \psi_L$ for graphs encoding words of $L = (a \uparrow b \uparrow c \uparrow) \ast \cup (A \ast (a b c) A \ast)$. We now have to rule out other graphs, in a monotone way:

$\psi -$ is a conjunction of edge requirements:
- $x a, x b, x c$ are at least linked as in the example,
- other vertices are always linked by an edge,...

$\psi +$ is a disjunction of excess edges:
- $x a x b$, ...

Final Formula: $\exists x a, x b, x c. (\psi - \land (\psi_L \lor \psi +))$

Left as exercise: Same with undirected graphs.
From finite words to finite graphs

Encode words into (directed) graphs, here $ab(a)\, c$: 

Final Formula: $\exists x^a, x^b, x^c. (\psi^- \land (\psi_{L} \lor \psi^+))$ 

Left as exercise: Same with undirected graphs.
From finite words to finite graphs

Encode words into (directed) graphs, here $ab^a_c$:

$$\rightarrow \text{formula } \psi_L \text{ for graphs encoding words of } L = (a^b c^c)^* \cup (A* \begin{pmatrix} a \\ b \\ c \end{pmatrix} A^*).$$
From finite words to finite graphs

Encode words into (directed) graphs, here $ab(a\uparrow b\uparrow)c$:

$\rightarrow$ formula $\psi_L$ for graphs encoding words of $L = (a\uparrow b\uparrow c\uparrow)^* \cup (a^* \left( \begin{array}{c} a \\ b \\ c \end{array} \right) A^*)$.

We now have to rule out other graphs, in a monotone way:

$\blacktriangleright$ $\psi^-$ is a conjunction of edge requirements:
From finite words to finite graphs

Encode words into (directed) graphs, here $ab(c)$:

\[ \psi_L \]

→ formula $\psi_L$ for graphs encoding words of $L = (a \uparrow b \uparrow c) \ast \cup (A^* \left( \begin{array}{c} a \\ b \\ c \end{array} \right) A^*)$.

We now have to rule out other graphs, in a monotone way:

- $\psi^-$ is a conjunction of edge requirements:
  - $x_a, x_b, x_c$ are at least linked as in the example,
From finite words to finite graphs

Encode words into (directed) graphs, here $ab(a^b)c$:

$\rightarrow$ formula $\psi_L$ for graphs encoding words of $L = (a^b b^c c^a)^* \cup (A^* \left( \begin{array}{c} a \\ b \\ c \end{array} \right) A^*)$.

We now have to rule out other graphs, in a monotone way:

- $\psi^-$ is a conjunction of edge requirements:
  - $x_a, x_b, x_c$ are at least linked as in the example,
  - other vertices are always linked by an edge,...
From finite words to finite graphs

Encode words into (directed) graphs, here $ab(a\uparrow b)c$:

$\rightarrow$ formula $\psi_L$ for graphs encoding words of $L = (a\uparrow b\uparrow c\uparrow)^* \cup (A^* \binom{a}{b} A^*)$.

We now have to rule out other graphs, in a monotone way:

- $\psi^-$ is a conjunction of edge requirements:
  - $x_a, x_b, x_c$ are at least linked as in the example,
  - other vertices are always linked by an edge,...

- $\psi^+$ is a disjunction of excess edges:
From finite words to finite graphs

Encode words into (directed) graphs, here $ab(b)\, c$:

$$\rightarrow \text{formula } \psi_L \text{ for graphs encoding words of } L = (a^{↑} b^{↑} c^{↑})^* \cup (A^* \left( \begin{array}{c} a \\ b \\ c \end{array} \right) A^*).$$

We now have to rule out other graphs, in a **monotone** way:

- $\psi^-$ is a conjunction of **edge requirements**:
  - $x_a, x_b, x_c$ are at least linked as in the example,
  - other vertices are always linked by an edge, . . .

- $\psi^+$ is a disjunction of **excess edges**:
  - $x_a \rightarrow x_b$, . . .
From finite words to finite graphs

Encode words into (directed) graphs, here \(ab\)\(^a\)\(_b\)\(c\):

\[
\begin{array}{c}
x_a \\
\rightarrow \\
x_b \\
\rightarrow \\
x_c \\
\end{array}
\]

→ formula \(\psi_L\) for graphs encoding words of \(L = (a^b c^c)^* \cup (A^* (\begin{array}{c} a \\ b \\ c \end{array}) A^*)\).

We now have to rule out other graphs, in a monotone way:

- \(\psi^-\) is a conjunction of edge requirements:
  - \(x_a, x_b, x_c\) are at least linked as in the example,
  - other vertices are always linked by an edge,…

- \(\psi^+\) is a disjunction of excess edges:
  - \(x_a \xrightarrow{} x_b\),
  - ,
  - ,…

\[\exists x_a, x_b, x_c. (\psi^- \land (\psi_L \lor \psi^+))\]

Left as exercise: Same with undirected graphs.
From finite words to finite graphs

Encode words into (directed) graphs, here \( ab^\left(\begin{array}{c} a \\ b \end{array}\right) c \):

\[
\begin{array}{c}
\xrightarrow{x_a} & \xrightarrow{x_b} & \xrightarrow{x_c} \\
\bigcirc & \bigcirc & \bigcirc \\
\rightarrow & \rightarrow & \rightarrow \\
\bigbox & \bigbox & \bigbox
\end{array}
\]

→ formula \( \psi_L \) for graphs encoding words of \( L = (a^\uparrow b^\uparrow c^\uparrow)^* \cup (A^* \left(\begin{array}{c} a \\ b \\ c \end{array}\right) A^*) \).

We now have to rule out other graphs, in a monotone way:

- \( \psi^- \) is a conjunction of edge requirements:
  - \( x_a, x_b, x_c \) are at least linked as in the example,
  - other vertices are always linked by an edge,…

- \( \psi^+ \) is a disjunction of excess edges:
  - \( x_a \xrightarrow{\bigcirc} x_b \),
  - \( \bigbox \xrightarrow{\bigbox} \bigbox \),…

**Final Formula:** \( \exists x_a, x_b, x_c. (\psi^- \land (\psi_L \lor \psi^+)) \)
From finite words to finite graphs

Encode words into (directed) graphs, here $ab(a \uparrow b) c$:

\[
\begin{array}{c}
   \xrightarrow{x_a} \xrightarrow{x_b} \xrightarrow{x_c} \\
   \xrightarrow{} \xrightarrow{} \xrightarrow{}
\end{array}
\]

$\rightarrow$ formula $\psi_L$ for graphs encoding words of $L = (a^\uparrow b^\uparrow c^\uparrow)^* \cup (A^* \left( \begin{array}{c} a \\ b \\ c \end{array} \right) A^*)$.

We now have to rule out other graphs, in a monotone way:

$\psi^-$ is a conjunction of edge requirements:

$\checkmark$ $x_a, x_b, x_c$ are at least linked as in the example,
$\checkmark$ other vertices are always linked by an edge,\ldots

$\psi^+$ is a disjunction of excess edges:

$\checkmark$ $x_a \rightarrow x_b$,
$\checkmark$ $\xrightarrow{} \xrightarrow{}, \ldots$

**Final Formula:** $\exists x_a, x_b, x_c. (\psi^- \land (\psi_L \lor \psi^+))$

*Left as exercise:* Same with undirected graphs.
Back to regular languages

Theorem
Given $L$ regular on an ordered alphabet, it is decidable whether

- $L$ is monotone (e.g. automata inclusion)
- $L$ is FO-definable [Schützenberger, McNaughton, Papert]

Can we decide whether $L$ is FO$^+$-definable?
Theorem

Given $L$ regular on an ordered alphabet, it is **decidable** whether

- $L$ is monotone (e.g. automata inclusion)
- $L$ is $FO$-definable [Schützenberger, McNaughton, Papert]

Can we decide whether $L$ is $FO^+$-definable?

Theorem

$FO^+$-definability is **undecidable** for regular languages.
Back to regular languages

**Theorem**

*Given* $L$ *regular on an ordered alphabet, it is decidable whether*

- $L$ *is monotone* (e.g. automata inclusion)
- $L$ *is* $\text{FO}$-definable [Schützenberger, McNaughton, Papert]

*Can we decide whether $L$ is $\text{FO}^+$-definable?*

**Theorem**

$\text{FO}^+$-definability is undecidable for regular languages.

Reduction from *Turing Machine Mortality*:

A deterministic TM $M$ is *mortal* if there a uniform bound $n$ on the runs of $M$ from *any* configuration.

Undecidable [Hooper 1966].
Undecidability proof sketch

Given a TM $M$, we build a regular language $L$ such that

$$M \text{ mortal } \iff L \text{ is } \text{FO}^+\text{-definable.}$$
Undecidability proof sketch

Given a TM $M$, we build a regular language $L$ such that

$$ M \text{ mortal } \iff L \text{ is } \text{FO}^+\text{-definable.} $$

**Building $L$:**

Inspired from $(a^\uparrow b^\uparrow c^\uparrow)^*$, but:

- $a, b, c \rightsquigarrow$ Words from languages $C_1, C_2, C_3$ encoding configs of $M$.

- All transitions of $M$ follow the cycle:

$$ C_1 \leftarrow C_2 \rightarrow C_3 $$

- $(a^\uparrow b^\uparrow c^\uparrow) \rightsquigarrow (u_1^\uparrow u_2^\uparrow)$, exists iff $u_1 \xrightarrow{M} u_2$. 
Undecidability proof sketch

Given a TM $M$, we build a regular language $L$ such that

$$M \text{ mortal } \iff L \text{ is } \text{FO}^{+}-\text{definable}.$$ 

**Building $L$:**

Inspired from $(a^\uparrow b^\uparrow c^\uparrow)^*$, but:

- $a, b, c \leadsto$ Words from languages $C_1, C_2, C_3$ encoding configs of $M$.

- All transitions of $M$ follow the cycle:

  $C_1 \xleftarrow{\cdot} C_2 \xrightarrow{\cdot} C_3$

- $(\binom{a}{b}, \binom{b}{c}, \binom{c}{a}) \leadsto (\frac{u_1}{u_2})$, exists iff $u_1 \xrightarrow{M} u_2$.

We choose

$$L := (C_1^\uparrow \cdot C_2^\uparrow \cdot C_3^\uparrow)^*$$
Undecidability proof sketch

Given a TM $M$, we build a regular language $L$ such that

\[ M \text{ mortal } \iff L \text{ is } \text{FO}^+\text{-definable}. \]

**Building $L$:**

Inspired from $(a^\uparrow b^\uparrow c^\uparrow)^*$, but:

- $a, b, c \leadsto$ Words from languages $C_1, C_2, C_3$ encoding configs of $M$.

- All transitions of $M$ follow the cycle:

\[
\begin{array}{c}
\uparrow & \downarrow \\
C_1 & C_2 & C_3 \\
\downarrow & \uparrow
\end{array}
\]

- $(a^b), (b^c), (c^a) \leadsto (u_1^u_2)$, exists iff $u_1 \xrightarrow{M} u_2$.

We choose

\[ L := (C_1^\uparrow \cdot C_2^\uparrow \cdot C_3^\uparrow)^* \]

⚠️ $u \in L \not\Rightarrow u$ encodes a run of $M$. 

16/18
The reduction

If $M$ not mortal:
Let $u_1, u_2, \ldots, u_n$ a long run of $M$, and play Duplicator in :

\[
\begin{align*}
\forall u \in L : & \quad u_1 \; u_2 \; u_3 \; \ldots \; u_{n-1} \; u_n \\
\forall v \not\in L : & \quad (u_1 \; u_2) \; (u_2 \; u_3) \; (u_3 \; u_4) \; \ldots \; (u_{n-1} \; u_n)
\end{align*}
\]

$\rightarrow L$ is not $\text{FO}^+$-definable.
The reduction

If $M$ not mortal:
Let $u_1, u_2, \ldots, u_n$ a long run of $M$, and play Duplicator in :

$$u \in L : u_1 u_2 u_3 \ldots u_{n-1} u_n$$

$$v \notin L : (u_1) (u_2) (u_3) \ldots (u_{n-1})$$

$\rightarrow L$ is not $\text{FO}^+$-definable.

If $M$ mortal with bound $n$:
Abstract $u_i$ by the length of the run of $M$ starting in it (at most $n$).
The reduction

If $M$ not mortal:
Let $u_1, u_2, \ldots, u_n$ a long run of $M$, and play Duplicator in:

$$u \in L: \ u_1\ u_2\ u_3\ \ldots\ u_{n-1}\ u_n$$

$$v \notin L: \ (\frac{u_1}{u_2})\ (\frac{u_2}{u_3})\ (\frac{u_3}{u_4})\ \ldots\ (\frac{u_{n-1}}{u_n})$$

$\rightarrow L$ is not $\text{FO}^+$-definable.

If $M$ mortal with bound $n$:
Abstract $u_i$ by the length of the run of $M$ starting in it (at most $n$).

Play Spoiler in the abstracted game (here $n = 5$):

$$u: \ 2\ 3\ 2\ 4\ 3\ 5\ 4\ 3\ 4\ 4$$

$$v: \ \frac{2}{1}\ \frac{3}{2}\ \frac{2}{1}\ \frac{4}{3}\ \frac{3}{2}\ \frac{5}{4}\ \frac{4}{3}\ \frac{5}{4}\ \frac{5}{4}$$

Spoiler always wins in $2^n$ rounds $\rightarrow L$ is $\text{FO}^+$-definable.
The reduction

If $M$ not mortal:
Let $u_1, u_2, \ldots, u_n$ a long run of $M$, and play Duplicator in :

\[
\begin{align*}
    u & \in L : & u_1 & u_2 & u_3 & \ldots & u_{n-1} & u_n \\
    \nu \notin L : & (u_1) & (u_2) & (u_3) & \ldots & (u_{n-1}) & (u_n)
\end{align*}
\]

$\rightarrow L$ is not $\text{FO}^+$-definable.

If $M$ mortal with bound $n$:
Abstract $u_i$ by the length of the run of $M$ starting in it (at most $n$).

Play Spoiler in the abstracted game (here $n = 5$):

\[
\begin{align*}
    u : & 2 & 3 & 2 & 4 & 3 & 5 & 4 & 3 & 4 & 4 \\
    \nu : & (2 \ 1) & (3 \ 2) & (2 \ 1) & (4 \ 3) & (3 \ 2) & (5 \ 4) & (4 \ 3) & (5 \ 4) & (5 \ 4)
\end{align*}
\]

Spoiler always wins in $2n$ rounds $\rightarrow L$ is $\text{FO}^+$-definable.
Ongoing work

With Thomas Colcombet:
Exploring the consequences of this in other frameworks:

▶ regular cost functions,
▶ logics on linear orders,
▶ ...

Slogan:
FO variants without negation will often display this behaviour.
Ongoing work

With Thomas Colcombet:
Exploring the consequences of this in other frameworks:

▶ regular cost functions,
▶ logics on linear orders,
▶ ... 

Slogan:
FO variants without negation will often display this behaviour.

Thanks for your attention!