Positive and monotone fragments of FO and LTL

Denis Kuperberg, Quentin Moreau

CNRS, LIP, ENS Lyon, Plume Team

ATLAS, 23 April 2024
First-Order Logic (FO)

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

**Syntax** of FO:

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

Graph class: Cliques

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

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

Example graph

Model of \(\varphi\)

Model of \(\psi\)
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 \( \neg \)

**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**: \( \phi \) 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 $\neg$

**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 ¬

**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 \( \neg \)

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?
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: FO-definable + monotone $\Rightarrow$ FO-definable without $\neg$. 
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 \textbf{infinite} structures.
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...
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:

- [Ajtaı, 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

- [K. 2021, 2023]
  EF games on words, elementary
FO on words, the usual way

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

$x \leq y$ 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 $(\leq, a, b[, \ldots ])$

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

- $x \leq y$ 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 $A^* aA^* bA^*$.
- $\neg a(x) \equiv \bigvee_{\beta \neq a} \beta(x)$. 
FO on words, the usual way

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

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

- $x \leq y$ 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 $A^* a A^*$.
- $\exists x, y . (x \leq y \land a(x) \land b(y))$. Language $A^* a A^* b A^*$.
- $\neg a(x) \equiv \bigvee_{\beta \neq a} \beta(x)$.

Theorem

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

Example: $(aa)^*$ is not FO-definable.
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 \( \iff \) LTL \( \iff \) counter-free automaton \( \iff \) ...
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 $\forall u, v, w \in A^*$:

$$uv^nw \in L \iff uv^{n+1}w \in L.$$
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 $\forall u, v, w \in A^*$:

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

$\iff$ 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.

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

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

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

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

- FO languages on alphabet \(A\) are the same \((\text{Preds}=\Sigma\) or \(A\)).
For now, a word is a structure \((X, \leq, a, b, \ldots)\) where

\[
\begin{align*}
\leq & \text{ is a total order} \\
(a, b, \ldots) & \text{ form a partition of } X.
\end{align*}
\]

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

\[
\text{Words on alphabet } \mathcal{P}(\{a, b, \ldots\}): \quad \emptyset \quad \{b\} \quad \{a, b\} \quad \emptyset \quad \{b\}
\]

\[
\bullet \quad \rightarrow \quad \bullet \quad \rightarrow \quad \bullet \quad \rightarrow \quad \bullet \quad \rightarrow \quad \bullet
\]

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

\[
\begin{align*}
\text{Useful e.g. in verification (LTL, \ldots):} \\
&\text{independent signals can be true or false simultaneously.}
\end{align*}
\]

\[
\text{FO languages on alphabet } A \text{ are the same (Preds=}\Sigma\text{ or } A). \\
\text{We no longer have } -a(x) \equiv \bigvee_{\beta \neq a} \beta(x).
\]
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 \(\mathcal{P}(\{a, b, \ldots\})\):

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

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 \(-a(x) \equiv \bigvee_{\beta \neq a} \beta(x)\).
  \(\rightarrow\) Negation necessary for full FO.
The $\text{FO}^+$ logic: positive formulas

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

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

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

More generally: $\text{FO}^+$ 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]: $\text{FO} \& \text{monotone} \Rightarrow \text{FO}^+$
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 $\Sigma = \{a, b\}$:

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

\[\exists x, y. (x \leq y) \land a(x) \land b(y) \rightsquigarrow (A^*\{a\}A^*\{b\}A^*) \cup (A^*\{a, b\}A^*)\]

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

More generally: 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 $\Sigma = \{a, b\}$:

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

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

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

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

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 \( \Sigma = \{a, b\} \):

\[ \exists x, y. (x \leq y) \land a(x) \land b(y) \rightsquigarrow (A^*\{a\}A^*\{b\}A^*) \cup (A^*\{a, b\}A^*) \]

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]: FO & monotone \( \Rightarrow \text{FO}^+ \)
A counter-example language

**Theorem [K. 2021,2023]**

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

Theorem [K. 2021,2023]

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 $a^\uparrow = \{a, (a_b), (c_a)\}$. 
A counter-example language

Theorem [K. 2021, 2023]

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

Alphabet $A = \{\emptyset, a, b, c, (\frac{a}{b}), (\frac{b}{c}), (\frac{c}{a}), (\frac{a}{b}c), (\frac{b}{c}a), (\frac{c}{a}b)\}$. Let $a^\uparrow = \{a, (\frac{a}{b}), (\frac{c}{a})\}$.

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

Theorem [K. 2021,2023]

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

Alphabet $A = \{\emptyset, a, b, c, (\frac{a}{b}), (\frac{b}{c}), (\frac{c}{a}), (\frac{a}{b}c), (\frac{b}{c}a), (\frac{c}{a}b)\}$. Let $a^\uparrow = \{a, (\frac{a}{b}), (\frac{b}{c})\}$.

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

**Theorem [K. 2021, 2023]**

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 \( v^{\geq 2} \))
A counter-example language

**Theorem [K. 2021,2023]**

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

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

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

**Lemma:** \( L \) is FO-definable.

**Proof:** \( a^{\uparrow} \) and \( b^{\uparrow} \) is counter-free. (no cycle labelled \( v \geq 2 \))

To prove \( L \) is not FO\(^+\)-definable: Ehrenfeucht-Fraïssé games.
Can we decide membership?

**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?
Can we decide membership?

**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 [K. 2021, 2023]**

$FO^+$-definability is **undecidable** for regular languages.
Can we decide membership?

**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 [K. 2021, 2023]**

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].
Corollaries: lifting the counter-example

\[ \text{Monotone-FO} \neq \text{FO}^+, \text{ and FO}^+ \text{ membership undecidable} \]
in the following settings:

- Finite graphs, edge predicate [K. 2023]
- Finite structures, arbitrary predicates [K. 2021, 2023]
- Words indexed by linear order, finiteness predicate
- Cost functions on finite words, boundedness predicate
Corollaries: lifting the counter-example

\[ \text{Monotone-FO} \neq \text{FO}^+, \text{and FO}^+ \text{ membership undecidable} \]
in the following settings:

- Finite graphs, edge predicate [K. 2023] New
- Finite structures, arbitrary predicates [K. 2021, 2023]
  simpler than [Ajtai Gurevich 1987, Stolboushkin 1995]
- Words indexed by linear order, finiteness predicate New
- Cost functions on finite words, boundedness predicate
  contradicts [K. 2011, 2014]
From finite words to finite graphs

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

\[
x_a \xrightarrow{} x_b \xrightarrow{} x_c
\]

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

Final Formula:

\[
\exists x_a \land ( \psi^- \lor (\psi^L \lor \psi^+))
\]

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

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

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

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

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

Rule out other graphs, in a **monotone** way:

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

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

$$
\begin{array}{c}
x_a \\
\downarrow \\
x_b \\
\downarrow \\
x_c \\
\end{array}
$$

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

Rule out other graphs, in a monotone way:

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

  $$
  \exists \begin{array}{c}
x_a \\
\downarrow \\
x_b \\
\downarrow \\
x_c \\
\end{array}
  $$

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

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

$\begin{array}{c}
\xrightarrow{\psi} \text{formula } \psi_L \text{ for graphs encoding words of } L = (a \uparrow b \uparrow c \uparrow)^* \cup (A^* \binom{a}{b} A^*). \\
\text{Rule out other graphs, in a monotone way:}
\end{array}$

$\begin{array}{c}
\quad \psi^- \text{ is a conjunction of edge requirements:}
\end{array}$

$\begin{array}{c}
\quad \exists \binom{x_a}{x_b} \xrightarrow{\psi^-} \binom{x_c}{x_a}
\end{array}$

$\begin{array}{c}
\quad \forall \square x, y. (x \rightarrow y) \lor (y \rightarrow x)
\end{array}$
Encode words into (directed) graphs, here $ab\left(\begin{smallmatrix}a \\ b \end{smallmatrix}\right)c$: 

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

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

Rule out other graphs, in a monotone way:

- $\psi^-$ is a conjunction of edge requirements:
  - $\exists x_a \rightarrow x_b \rightarrow x_c$
  - $\forall x, y. (x \rightarrow y) \lor (y \rightarrow x)$

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

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

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

Rule out other graphs, in a monotone way:

- $\psi^-$ is a conjunction of edge requirements:
  - $\exists x_a, x_b, x_c$
  - $\forall \Box x, y. (x \rightarrow y) \lor (y \rightarrow x)$

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

Encode words into (directed) graphs, here $ab\begin{vmatrix}a \\ b \end{vmatrix}c$:

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

Rule out other graphs, in a monotone way:

- $\psi^-$ is a conjunction of edge requirements:
  - $\exists$ $\begin{array}{c}x_a \\ x_b \\ x_c \end{array}$
  - $\forall \square x, y. (x \rightarrow y) \lor (y \rightarrow x)$

- $\psi^+$ is a disjunction of excess edges:
  - $\begin{array}{c}x_a \\ x_b \end{array}$,
  - $\square$, $\square$, $\ldots$
From finite words to finite graphs

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

$\psi_L$ for graphs encoding words of $L = (a^1b^1c^1)^* \cup (A^* \left( \frac{a}{b} \frac{b}{c} \right) A^*)$.

Rule out other graphs, in a monotone way:

- $\psi^-$ is a conjunction of edge requirements:
  - $\exists x_a, x_b, x_c$
  - $\forall \Box x, y.(x \to y) \lor (y \to x)$

- $\psi^+$ is a disjunction of excess edges:
  - $x_a \rightarrow x_b,$
  - \ldots

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 $ab \overset{a}{\underset{b}{\uparrow}} c$:

\[
\begin{array}{c}
\xymatrix{
  x_a \ar[r] & x_b \ar[r] & x_c \\
  \phantom{x} \ar[r] & \phantom{x} \ar[r] & \phantom{x}
}\end{array}
\]

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

Rule out other graphs, in a monotone way:

- $\psi^-$ is a conjunction of edge requirements:
  - $\exists \ x_a \rightarrow x_b \rightarrow x_c$
  - $\forall \ x, y. (x \rightarrow y) \lor (y \rightarrow x)$

- $\psi^+$ is a disjunction of excess edges:
  - $\xymatrix{\phantom{x} \ar[r] & \phantom{x}}$
  - $\phantom{x} \rightarrow \xymatrix{\phantom{x}}, \ldots$

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

*Left as exercise:* Same with undirected graphs.
Back to words: Link with LTL

LTL syntax:

\[ \varphi, \psi ::= \bot \mid T \mid a \mid \varphi \land \psi \mid \varphi \lor \psi \mid X\varphi \mid \varphi U \psi \mid \varphi R \psi \mid \neg \varphi. \]

UTL syntax:

\[ \varphi, \psi ::= \bot \mid T \mid a \mid \varphi \land \psi \mid \varphi \lor \psi \mid X\varphi \mid Y\varphi \mid P\varphi \mid F\varphi \mid H\varphi \mid G\varphi \mid \neg \varphi. \]
Back to words: Link with LTL

LTL syntax:

\[ \varphi, \psi ::= \bot | \top | a | \varphi \land \psi | \varphi \lor \psi | \text{X} \varphi | \varphi \text{U} \psi | \varphi \text{R} \psi | \neg \varphi. \]

UTL syntax:

\[ \varphi, \psi ::= \bot | \top | a | \varphi \land \psi | \varphi \lor \psi | \text{X} \varphi | \text{Y} \varphi | \text{P} \varphi | \text{F} \varphi | \text{H} \varphi | \text{G} \varphi | \neg \varphi. \]

Theorem

1. FO = LTL = FO$_3$ [Kamp 1968]
2. FO$_2[S, <] = UTL$ [Etessami, Vardi, Wilke 1997]
3. FO$_2[<] = UTL[P, F, G, H]$ [Etessami, Vardi, Wilke 1997]
Back to words: Link with LTL

LTL syntax:

\[ \varphi, \psi ::= \perp \mid T \mid a \mid \varphi \land \psi \mid \varphi \lor \psi \mid X\varphi \mid \varphi U \psi \mid \varphi R \psi \mid \neg \varphi. \]

UTL syntax:

\[ \varphi, \psi ::= \perp \mid T \mid a \mid \varphi \land \psi \mid \varphi \lor \psi \mid X\varphi \mid Y\varphi \mid P\varphi \mid F\varphi \mid H\varphi \mid G\varphi \mid \neg \varphi. \]

Theorem

- \( FO^+ = LTL^+ = FO_3^+ \) [K., Moreau]
- \( FO_2^+[S, <] = UTL^+ \) [K., Moreau]
- \( FO_2^+[<] = UTL^+[P, F, G, H] \) [K., Moreau]
Refining the counter-example language

What is needed to obtain $\text{FO}^+ \neq \text{FO} \cap \text{Monotone}$?
Refining the counter-example language

What is needed to obtain $\text{FO}^+ \neq \text{FO} \cap \text{Monotone}$?

**Theorem (K., Moreau)**

There is a counter-example language definable in

- $\text{FO}$ with one unary predicate (instead of 3)
- $\text{FO}[\text{between}] : \text{bet}(a,x,y)$ means $\exists z$ between $x$ and $y$ s.t. $a(z)$. 
Refining the counter-example language

What is needed to obtain \( \text{FO}^+ \neq \text{FO} \cap \text{Monotone} \)?

**Theorem (K., Moreau)**

There is a counter-example language definable in

- \( \text{FO} \) with one unary predicate (instead of 3)
- \( \text{FO}[^{\text{between}}] : \text{bet}(a,x,y) \text{ means } \exists z \text{ between } x \text{ and } y \text{ s.t. } a(z) \).

**Theorem (K., Moreau)**

There is no counter-example language definable in \( \text{FO}_2[<] \).
I.e. \( \text{FO}_2[<] \cap \text{Monotone} \subset \text{FO}^+ \).
Further work

Open problems:

- $\text{FO}_2 \cap \text{Monotone} \overset{?}{=} \text{FO}_2^+$
- For which fragments $F \subset \text{FO}$: $F \cap \text{Monotone} = F^+$
- Other kind of counterexamples?
Further work

Open problems:

- \( \text{FO}_2 \cap \text{Monotone} \stackrel{?}{=} \text{FO}_2^+ \)
- For which fragments \( F \subset \text{FO} \): \( F \cap \text{Monotone} = F^+ \)
- Other kind of counterexamples?

Thanks for your attention!