Tree Languages Definable with One Quantifier Alternation

Mikołaj Bojańczyk (Warszawa)
Luc Segoufin (Paris)

The following problem is decidable:

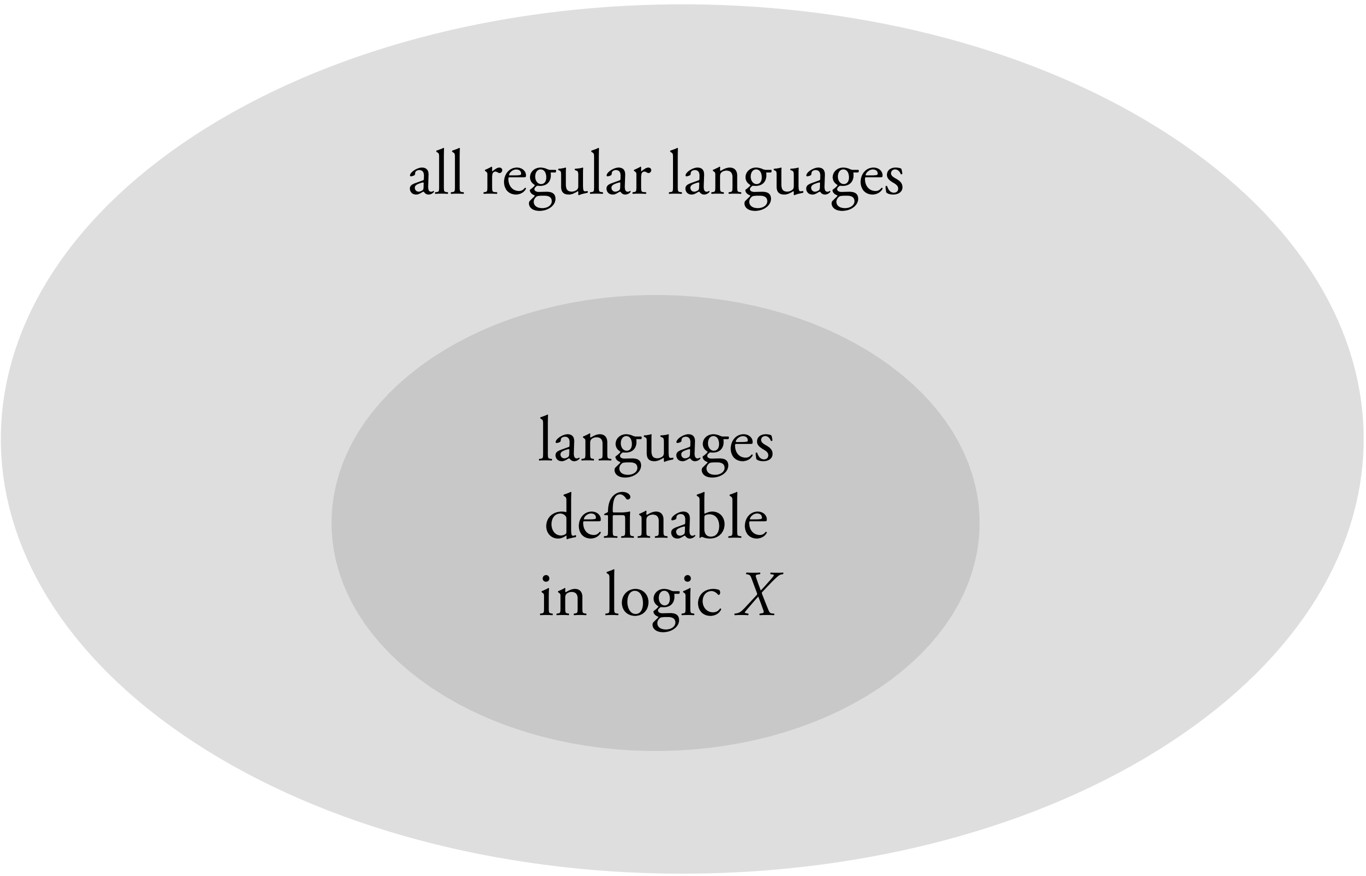
Input: A regular tree language L, given by a tree automaton.

Question: Is L definable by a formula with quantifier prefix $\exists^* \forall^*$ and also by a formula with quantifier prefix $\forall^*\exists^*$

This talk is about understanding the expressive power of logics on words and trees. The logics involved can only define (some) regular languages.

This talk is about understanding the expressive power of logics on words and trees. The logics involved can only define (some) regular languages.

Understand logic X =
give na algorithm to decide if a language L is definable in X



Why this notion of understanding?

There is a rich theory connecting logic, regular languages, and algebra.

Why this notion of understanding?

There is a rich theory connecting logic, regular languages, and algebra.

Theorem. (Schützenberger, McNaughton/Papert)
The following are equivalent for a word language:
– L is definable in first-order logic
– L is star-free
– the syntactic monoid of L is group-free

Why this notion of understanding?

There is a rich theory connecting logic, regular languages, and algebra.

Theorem. (Schützenberger, McNaughton/Papert)
The following are equivalent for a word language:
– L is definable in first-order logic
– L is star-free
– the syntactic monoid of L is group-free

... more results, including modulo quantifiers,
the quantifier alternation hierarchy, etc.

This paper is part of a program investigating the algebra-logic connection for trees. Eventually, we want to answer questions such as:

– what is the expressive power of first-order logic on trees?
– what is a tree group?
– is there a Krohn-Rhodes decomposition theory?

Why this notion of understanding?

Theorem. (Schützenberger, McNaughton/Papert)
The following are equivalent for a word language:

– L is definable in first-order logic
– L is star-free
– the syntactic monoid of L is group-free

... more results, including modulo quantifiers,
the quantifier alternation hierarchy, etc.

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)

The following formalisms define the same word languages:

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)
The following formalisms define the same word languages:

$$b^* \cdot a \cdot \{a,b,c\}^*$$

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)
The following formalisms define the same word languages:

$$b^* \cdot a \cdot \{a,b,c\}^*$$

1. Two-variable first-order logic

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)
The following formalisms define the same word languages:

$$b^* \cdot a \cdot \{a,b,c\}^*$$

1. Two-variable first-order logic

$$\exists x.\ a(x) \ \wedge \ \big(\forall y < x.\ b(y)\big)$$

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)
The following formalisms define the same word languages:

$$b^* \cdot a \cdot \{a,b,c\}^*$$

1. Two-variable first-order logic
2. Temporal logic with operators F and F^{-1}

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)
The following formalisms define the same word languages:

$$b^* \cdot a \cdot \{a,b,c\}^*$$

1. Two-variable first-order logic
2. Temporal logic with operators F and F^{-1}

$$\mathsf{F}\big(a \ \wedge\ \neg(\mathsf{F}^{-1}c)\big)$$

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)
The following formalisms define the same word languages:

$$b^* \cdot a \cdot \{a,b,c\}^*$$

1. Two-variable first-order logic
2. Temporal logic with operators F and F^{-1}
3. Languages definable with prefix $\exists^* \forall^*$
   and also with prefix $\forall^* \exists^*$

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)
The following formalisms define the same word languages:

$$b^* \cdot a \cdot \{a,b,c\}^*$$

1. Two-variable first-order logic
2. Temporal logic with operators F and F^{-1}
3. Languages definable with prefix $\exists^* \forall^*$
   and also with prefix $\forall^* \exists^*$

$$\forall x \exists y.\ c(x) \Rightarrow \big(y < x \wedge a(y)\big)$$

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)
The following formalisms define the same word languages:

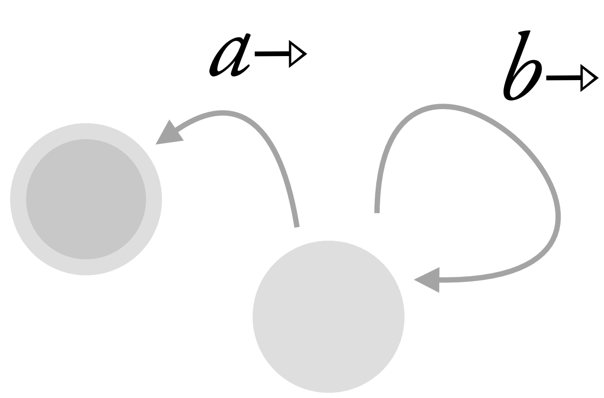
$$b^* \cdot a \cdot \{a,b,c\}^*$$

1. Two-variable first-order logic
2. Temporal logic with operators F and F^{-1}
3. Languages definable with prefix $\exists^* \forall^*$ and also with prefix $\forall^* \exists^*$
4. Two–way ordered deterministic automata

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)
The following formalisms define the same word languages:

$$b^* \cdot a \cdot \{a,b,c\}^*$$

1. Two-variable first-order logic
2. Temporal logic with operators F and F^{-1}
3. Languages definable with prefix $\exists^* \forall^*$ and also with prefix $\forall^* \exists^*$
4. Two–way ordered deterministic automata



Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)
The following formalisms define the same word languages:

$$b^* \cdot a \cdot \{a,b,c\}^*$$

1. Two-variable first-order logic
2. Temporal logic with operators F and F^{-1}
3. Languages definable with prefix $\exists^* \forall^*$ and also with prefix $\forall^* \exists^*$
4. Two–way ordered deterministic automata
5. Turtle automata

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)
The following formalisms define the same word languages:

$$b^* \cdot a \cdot \{a,b,c\}^*$$

1. Two-variable first-order logic
2. Temporal logic with operators F and F^{-1}
3. Languages definable with prefix $\exists^* \forall^*$
   and also with prefix $\forall^* \exists^*$
4. Two–way ordered deterministic automata
5. Turtle automata
   "go right to first *a;* go left to first *c*" fails
   "go right to first *a*" works

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)
The following formalisms define the same word languages:

$$b^* \cdot a \cdot \{a,b,c\}^*$$

1. Two-variable first-order logic
2. Temporal logic with operators F and F^{-1}
3. Languages definable with prefix $\exists^* \forall^*$ and also with prefix $\forall^* \exists^*$
4. Two–way ordered deterministic automata
5. Turtle automata
6. Monoids in DA

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)
The following formalisms define the same word languages:

$$b^* \cdot a \cdot \{a,b,c\}^*$$

1. Two-variable first-order logic
2. Temporal logic with operators F and F^{-1}
3. Languages definable with prefix $\exists^* \forall^*$
   and also with prefix $\forall^* \exists^*$
4. Two–way ordered deterministic automata
5. Turtle automata
6. Monoids in DA
7. A type of unambiguous expression

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)

The following formalisms define the same word languages:

$$b^* \cdot a \cdot \{a,b,c\}^*$$

1. Two-variable first-order logic
2. Temporal logic with operators F and F^{-1}
3. Languages definable with prefix $\exists^* \forall^*$ and also with prefix $\forall^* \exists^*$
4. Two–way ordered deterministic automata
5. Turtle automata
6. Monoids in DA
7. A type of unambiguous expression

What about trees?

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)
The following formalisms define the same word languages:

$$b^* \cdot a \cdot \{a,b,c\}^*$$

1. Two-variable first-order logic
2. Temporal logic with operators F and F^{-1}
3. Languages definable with prefix $\exists^* \forall^*$ and also with prefix $\forall^* \exists^*$
4. Two–way ordered deterministic automata
5. Turtle automata
6. Monoids in DA
7. A type of unambiguous expression

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)
The following formalisms define the same word languages:

$$b^* \cdot a \cdot \{a,b,c\}^*$$

1. Two-variable first-order logic
2. Temporal logic with operators F and F^{-1}
3. Languages definable with prefix $\exists^* \forall^*$
   and also with prefix $\forall^* \exists^*$
4. Two–way ordered deterministic automata
5. Turtle automata
6. Monoids in DA

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)
The following formalisms define the same word languages:

$$b^* \cdot a \cdot \{a,b,c\}^*$$

1. Two-variable first-order logic
2. Temporal logic with operators F and F^{-1}
3. Languages definable with prefix $\exists^* \forall^*$
   and also with prefix $\forall^* \exists^*$

5. Turtle automata
6. Monoids in DA

Theorem. (Etessami, Schützenberger, Schwentick, Thérien, Vollimer, Wilke)
The following formalisms define the same word languages:

$$b^* \cdot a \cdot \{a,b,c\}^*$$

1. Two-variable first-order logic
2. Temporal logic with operators F and F^{-1}
3. Languages definable with prefix $\exists^* \forall^*$
   and also with prefix $\forall^* \exists^*$

6. Monoids in DA

1. Two-variable first-order logic

2. Temporal logic with operators F and F^{-1}

3. Languages definable with prefix $\exists^* \forall^*$
   and also with prefix $\forall^* \exists^*$

6. Monoids in DA

1. Two-variable first-order logic

2. Temporal logic with operators F and F^{-1}

3. Languages definable with prefix $\exists^* \forall^*$
and also with prefix $\forall^* \exists^*$

6. Monoids in DA

1. Two-variable first-order logic

2. Temporal logic with operators F and F^{-1}

3. Languages definable with prefix $\exists^* \forall^*$
and also with prefix $\forall^* \exists^*$

6. Monoids in DA

1. Two-variable first-order logic

2. Temporal logic with operators F and F^{-1}

3. Languages definable with prefix $\exists^* \forall^*$
and also with prefix $\forall^* \exists^*$

6. Monoids in DA

1. Two-variable first-order logic

2. Temporal logic with operators F and F^{-1}

"two *a*'s"

3. Languages definable with prefix $\exists^* \forall^*$
and also with prefix $\forall^* \exists^*$

6. Monoids in DA

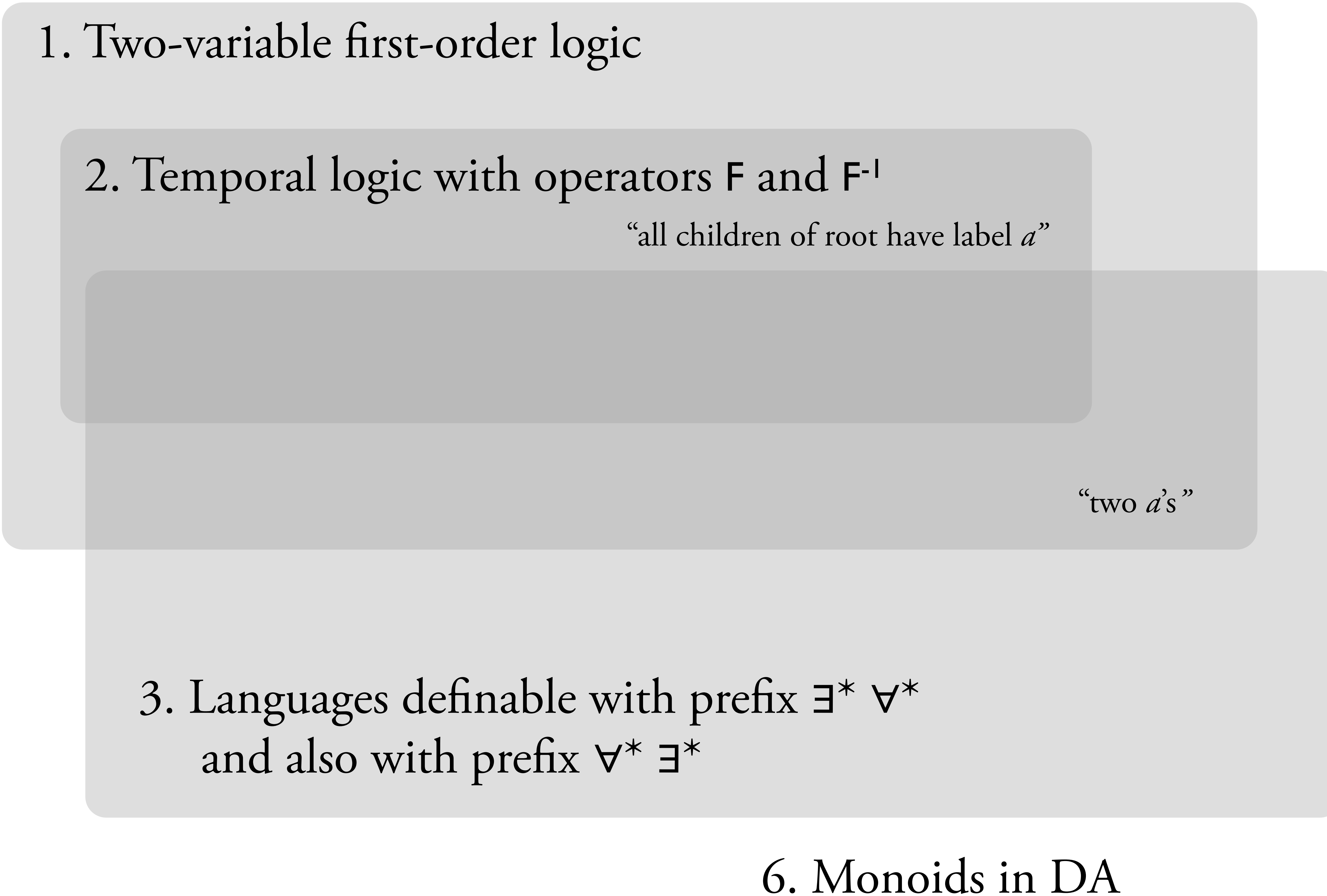
1. Two-variable first-order logic

2. Temporal logic with operators F and F^{-1}

"two *a*'s"

3. Languages definable with prefix $\exists^* \forall^*$
and also with prefix $\forall^* \exists^*$

6. Monoids in DA


1. Two-variable first-order logic
2. Temporal logic with operators F and F⁻¹
"all children of root have label *a*"
"two *a*'s"
3. Languages definable with prefix ∃* ∀*
and also with prefix ∀* ∃*
6. Monoids in DA

1. Two-variable first-order logic

2. Temporal logic with operators F and F^{-1}

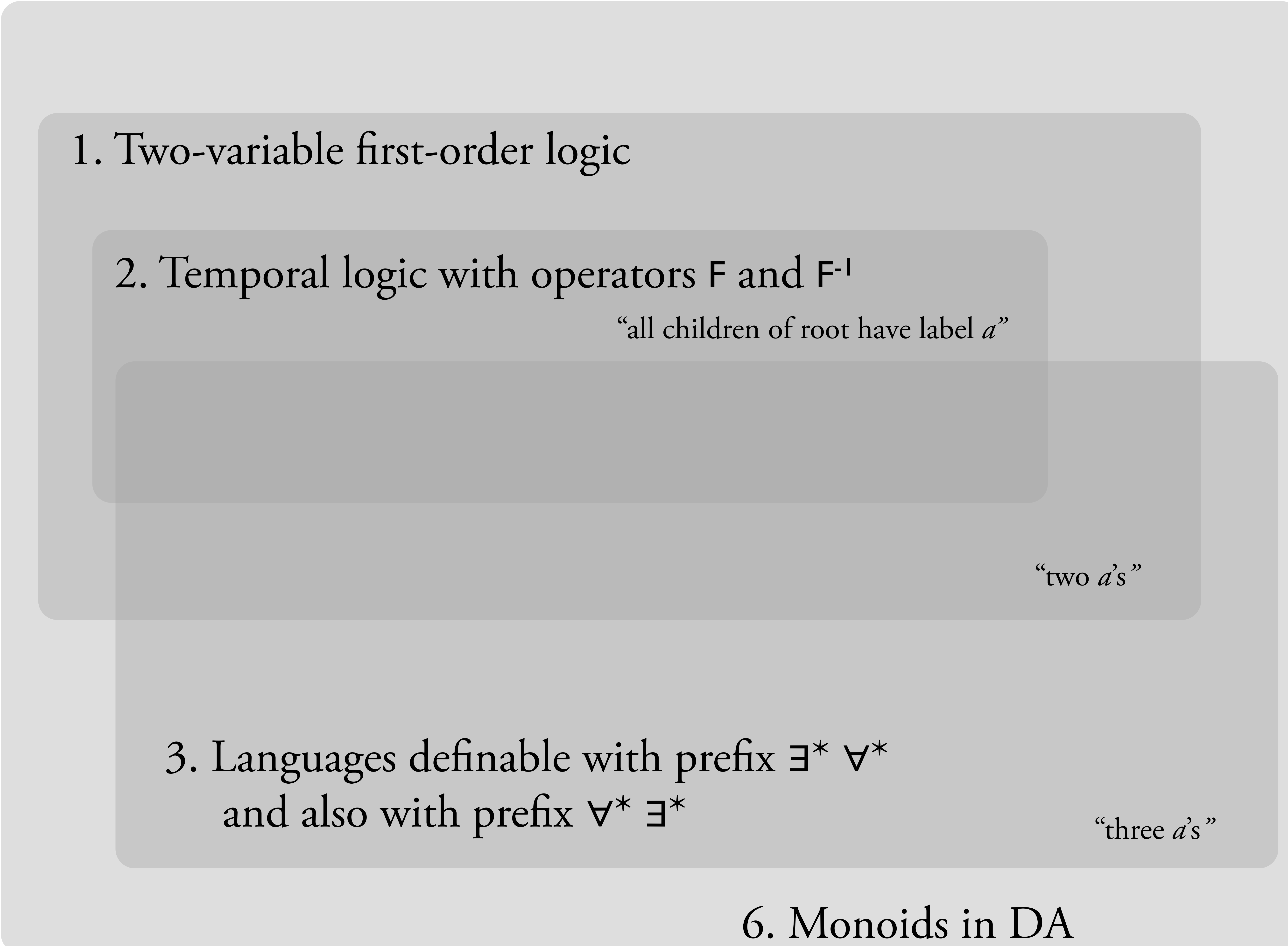
"all children of root have label *a*"

"two *a*'s"

3. Languages definable with prefix $\exists^* \forall^*$
and also with prefix $\forall^* \exists^*$

"three *a*'s"

6. Monoids in DA


1. Two-variable first-order logic
2. Temporal logic with operators F and F^{-1}
"all children of root have label *a*"
"two *a*'s"
3. Languages definable with prefix $\exists^* \forall^*$
and also with prefix $\forall^* \exists^*$
"three *a*'s"
6. Monoids in DA

1. Two-variable first-order logic

2. Temporal logic with operators F and F^{-1}

“all children of root have label *a*”

“two *a*’s”

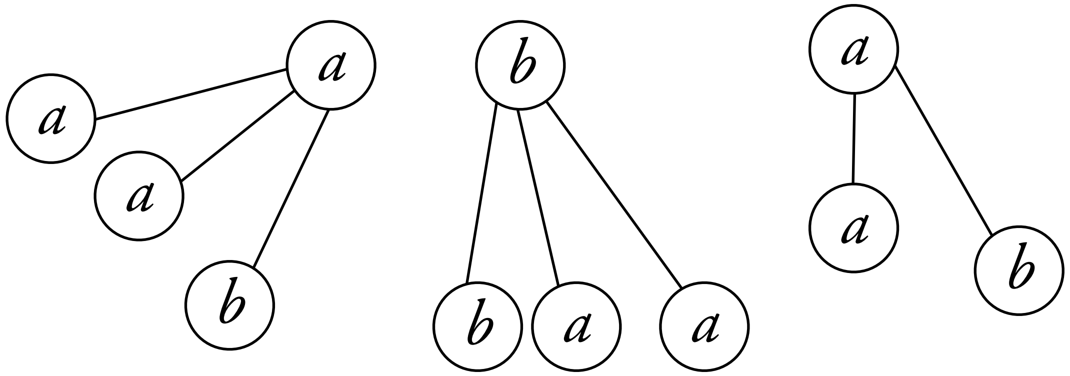
3. Languages definable with prefix $\exists^* \forall^*$
and also with prefix $\forall^* \exists^*$

“three *a*’s”

6. Monoids in DA

We consider forest languages
instead of tree languages

(a *forest* is a sequence of trees)



We use first-order formulas to describe properties of forests.
Variables quantify over nodes. Predicates allowed are:

"*x* ancestor of *y*" "*x* lexicographically before *y*" "label of *x* is *a*"

We are interested in forest languages that can be defined in both $\forall^* \exists^*$ and $\exists^* \forall^*$. We call this class Δ_2.

We are interested in forest languages that can be defined in both $\forall^* \exists^*$ and $\exists^* \forall^*$. We call this class Δ_2.

E.g. $\mathit{Trees} \in \Delta_2$

We are interested in forest languages that can be defined in both $\forall^* \exists^*$ and $\exists^* \forall^*$. We call this class Δ_2.

E.g. *Trees* $\in \Delta_2$

every two nodes have
a common ancestor

$$\forall x \forall y \exists z.\ z \leq x \wedge z \leq y$$

We are interested in forest languages that can be defined in both $\forall^* \exists^*$ and $\exists^* \forall^*$. We call this class Δ_2.

E.g. *Trees* $\in \Delta_2$

every two nodes have
a common ancestor

$$\forall x \forall y \exists z.\ z \leq x \wedge z \leq y$$

$\forall^* \exists^*$

We are interested in forest languages that can be defined in both $\forall^* \exists^*$ and $\exists^* \forall^*$. We call this class Δ_2.

E.g. *Trees* $\in \Delta_2$

every two nodes have a common ancestor

$$\forall x \forall y \exists z.\ z \leq x \wedge z \leq y$$

$\forall^* \exists^*$

some node is ancestor to every node

$$\exists x \forall y.\ x \leq y$$

We are interested in forest languages that can be defined in both $\forall^* \exists^*$ and $\exists^* \forall^*$. We call this class Δ_2.

E.g. *Trees* $\in \Delta_2$

every two nodes have a common ancestor

$$\forall x \forall y \exists z.\ z \leq x \wedge z \leq y$$

$\forall^* \exists^*$

some node is ancestor to every node

$$\exists x \forall y.\ x \leq y$$

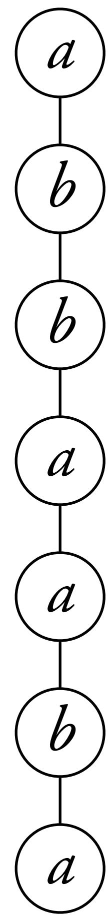
$\exists^* \forall^*$

Question:

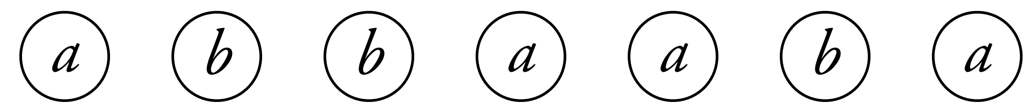
What forest languages can be defined in Δ_2?

Preferably, give an algorithm that decides if $L \in \Delta_2$.

a word w can have two encodings:



ver(w)

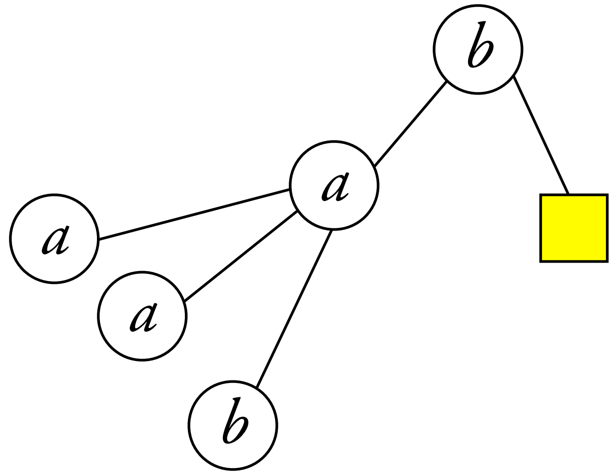


hor(w)

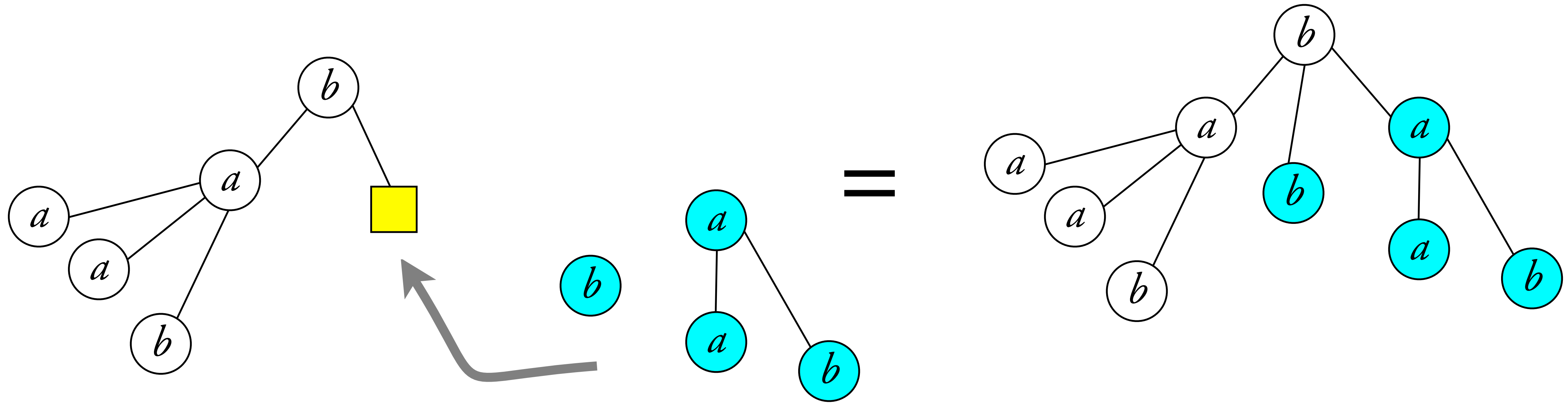
Fact. if L is a word language definable in Δ_2, then both hor(L) and ver(L) are forest languages definable in Δ_2.

Our characterization is stated as an identity. Intuitively, a forest language is definable in Δ_2 iff it admits a certain pumping lemma.

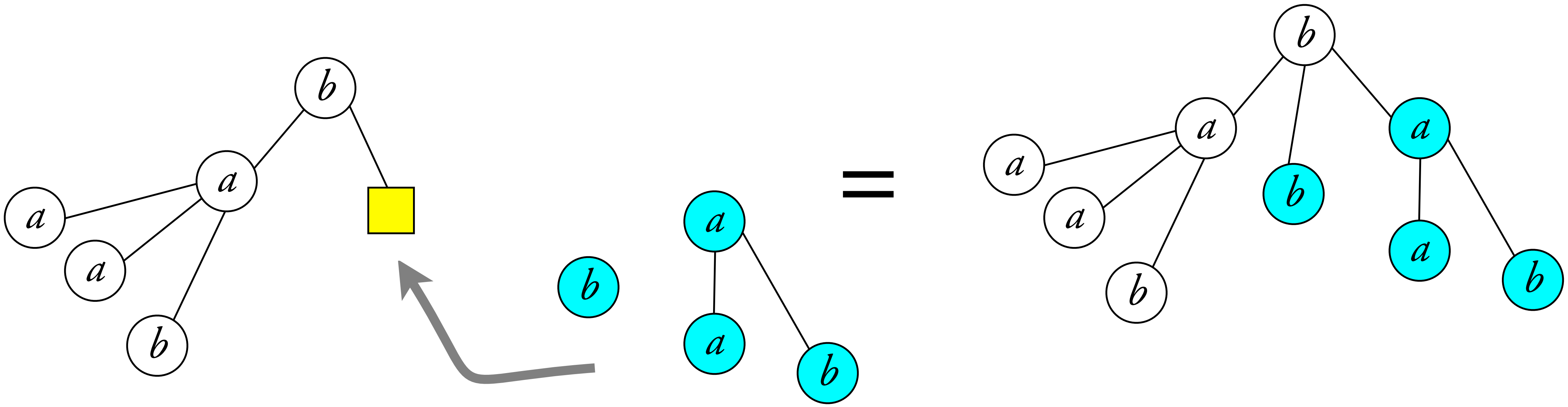
A *context* is a forest with a hole in a leaf



A *context* is a forest with a hole in a leaf



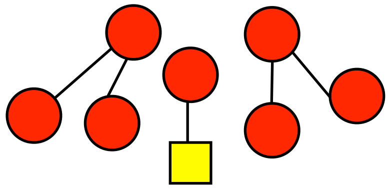
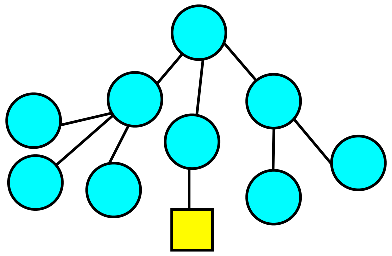
A *context* is a forest with a hole in a leaf



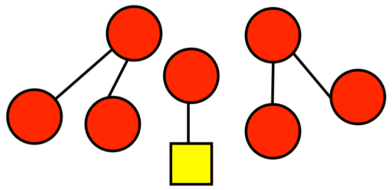
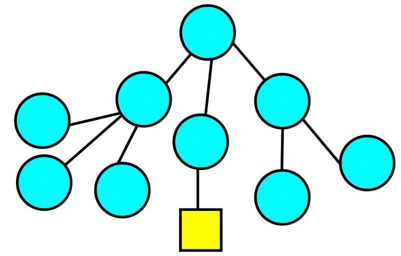
Notion of piece for contexts.

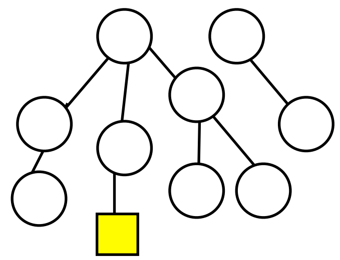
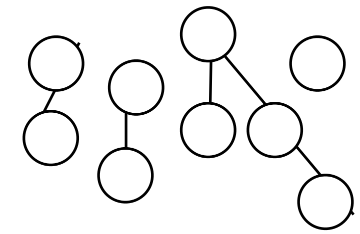
Myhill-Nerode congruence for a forest language L.

Myhill-Nerode congruence for a forest language L.

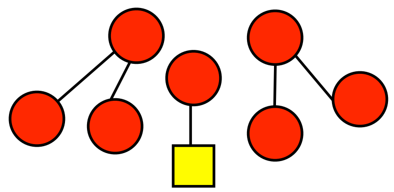
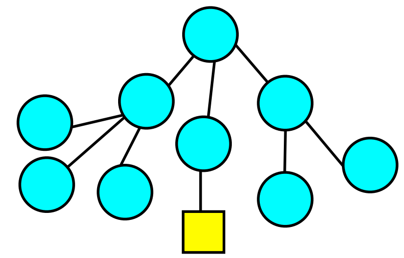
Two contexts  and  are called L-equivalent if

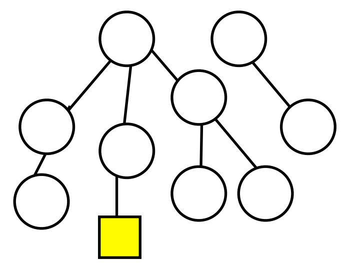
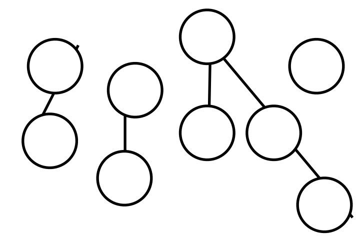
Myhill-Nerode congruence for a forest language L.

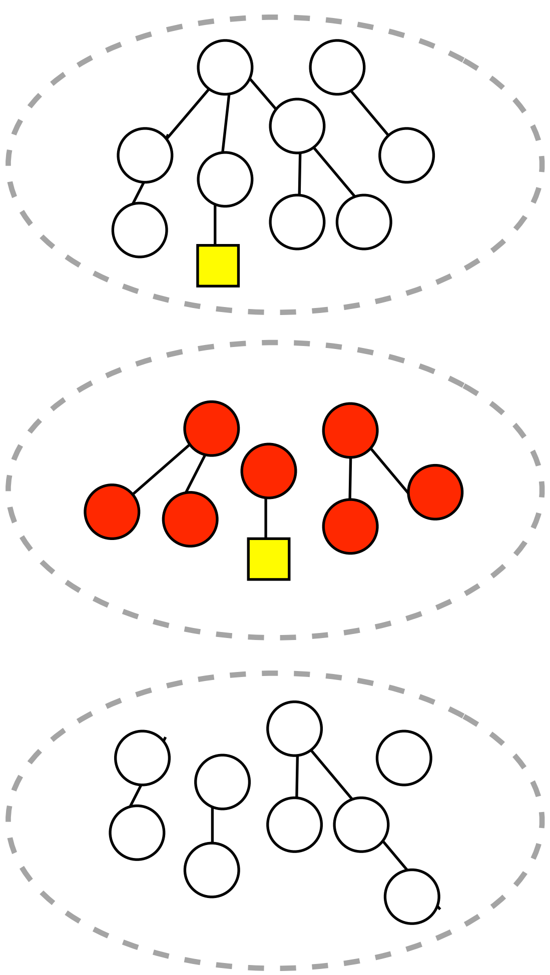
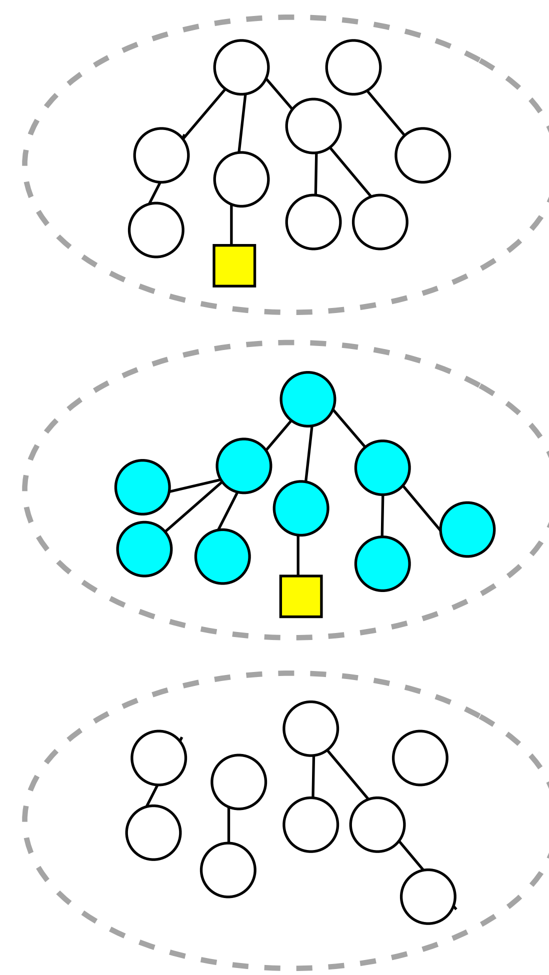
Two contexts  and  are called L-equivalent if

for every context  and every forest 

Myhill-Nerode congruence for a forest language L.

Two contexts  and  are called L-equivalent if

for every context  and every forest 

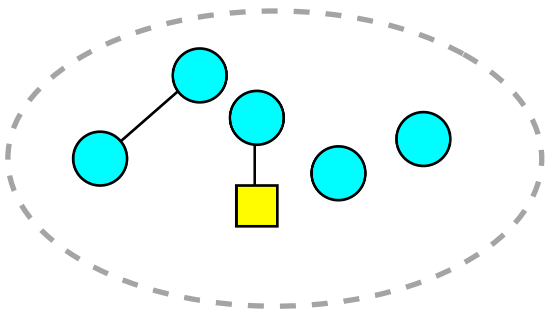
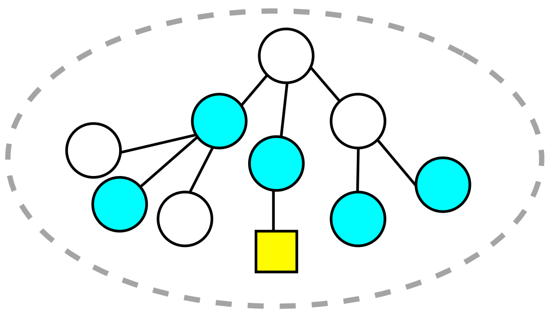
 $\in L$ iff  $\in L$

Main Theorem.

A forest language is definable in Δ_2 iff
    the following holds for all sufficiently large n

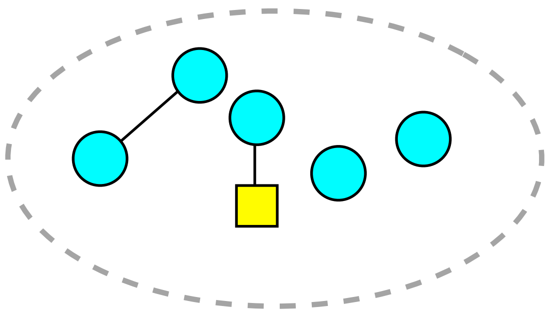
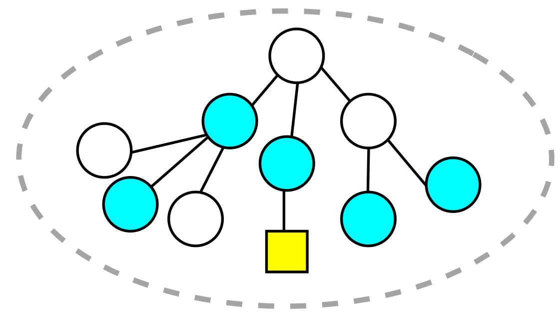
Main Theorem.

A forest language is definable in Δ_2 iff
the following holds for all sufficiently large n

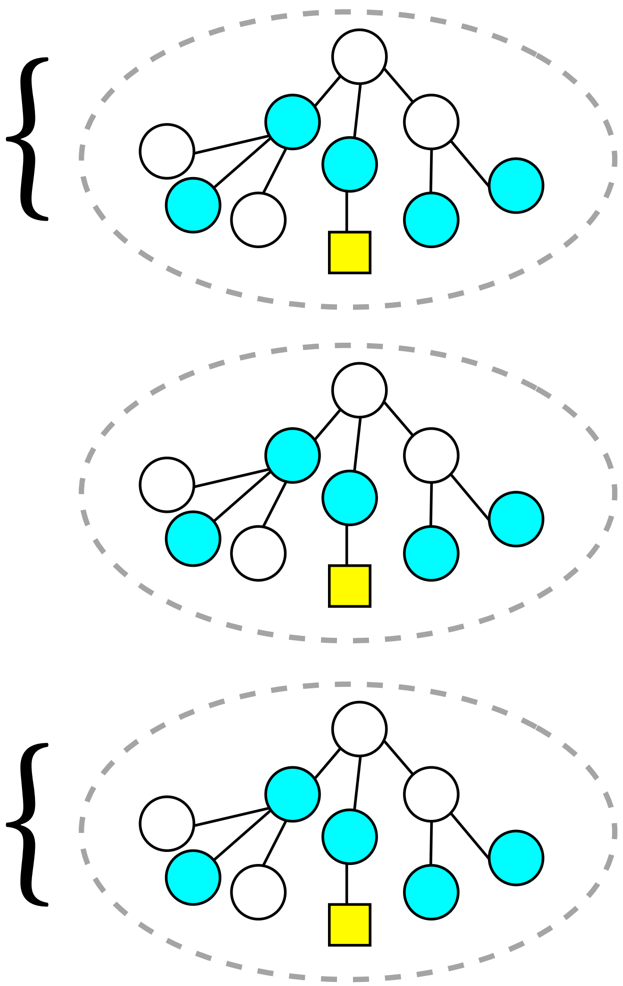
if  is a piece of  , then

Main Theorem.

A forest language is definable in Δ_2 iff
the following holds for all sufficiently large n

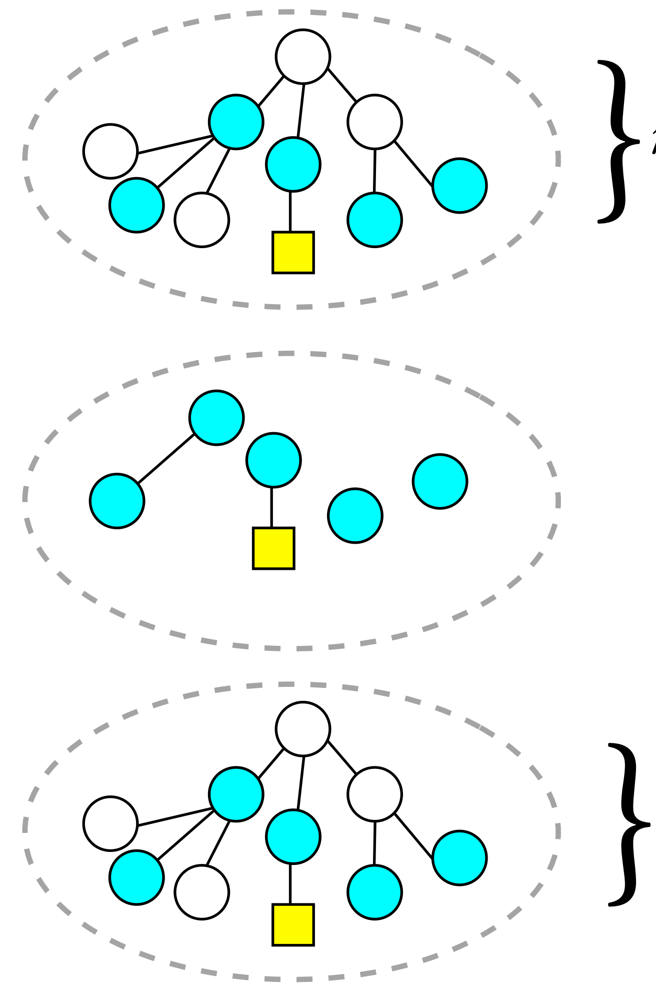
if  is a piece of  , then

n times {



n times {

is equivalent to



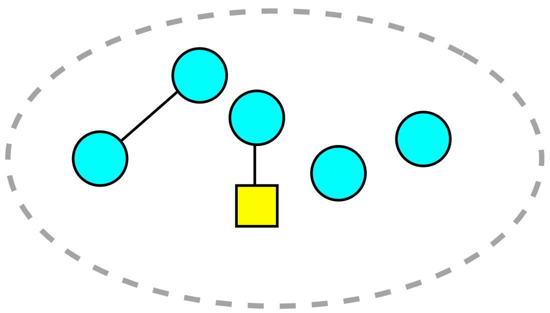
} n times

} n times

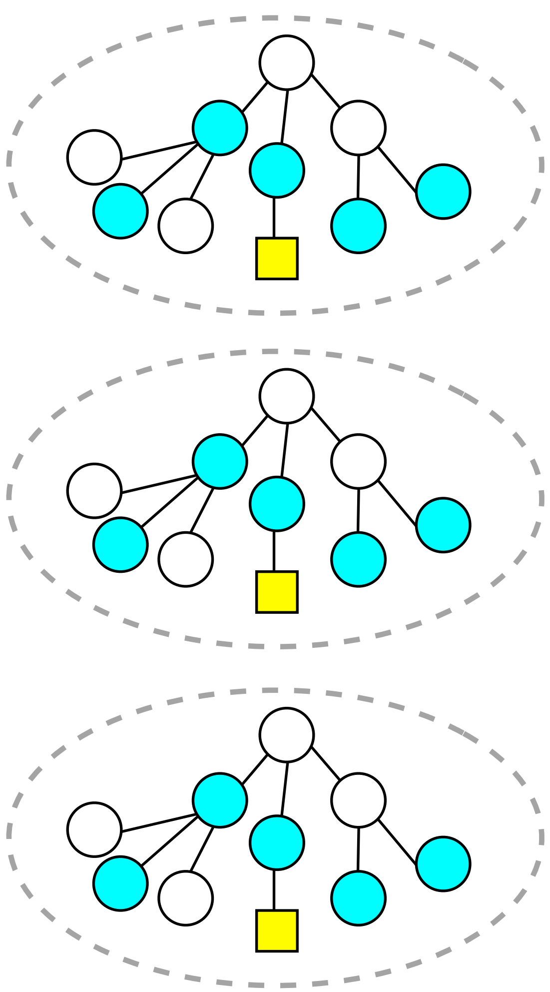
Main Theorem.

A forest language is definable in Δ_2 iff
the following holds for all sufficiently large n

This criterion is decidable.
We also have variants of the theorem for unordered trees / forests.

if  is a piece of , then

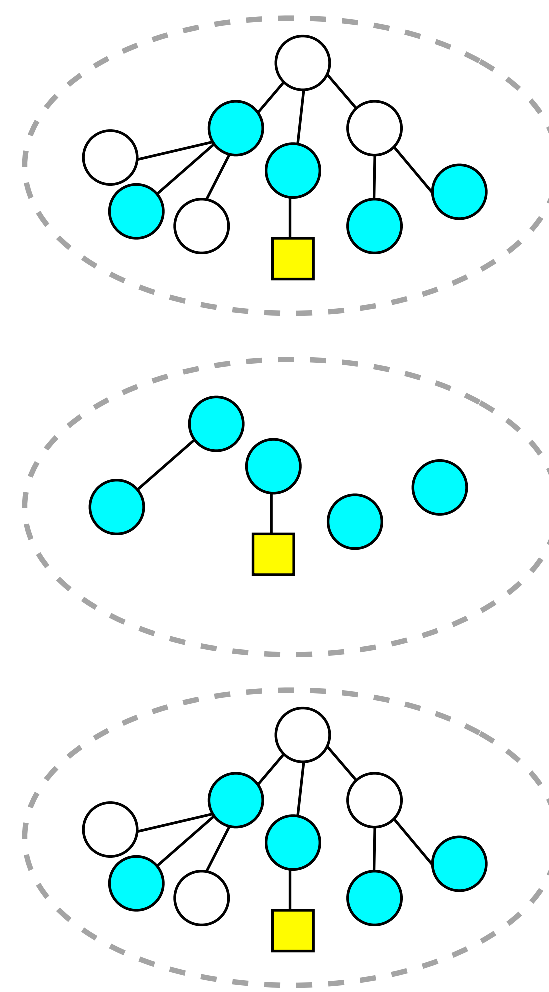
n times



n times

is equivalent to

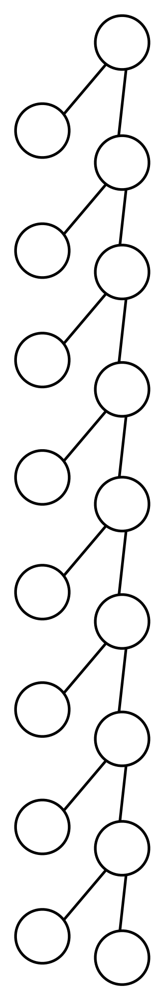
n times



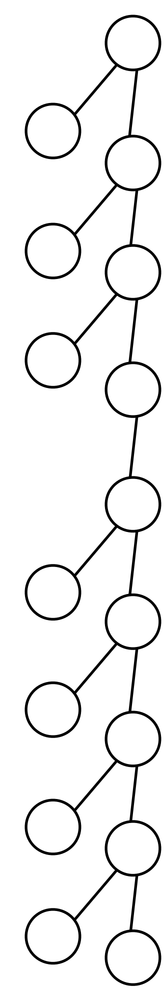
n times

Application.

The set of binary trees (every node has zero or two children) is not definable in Δ_2

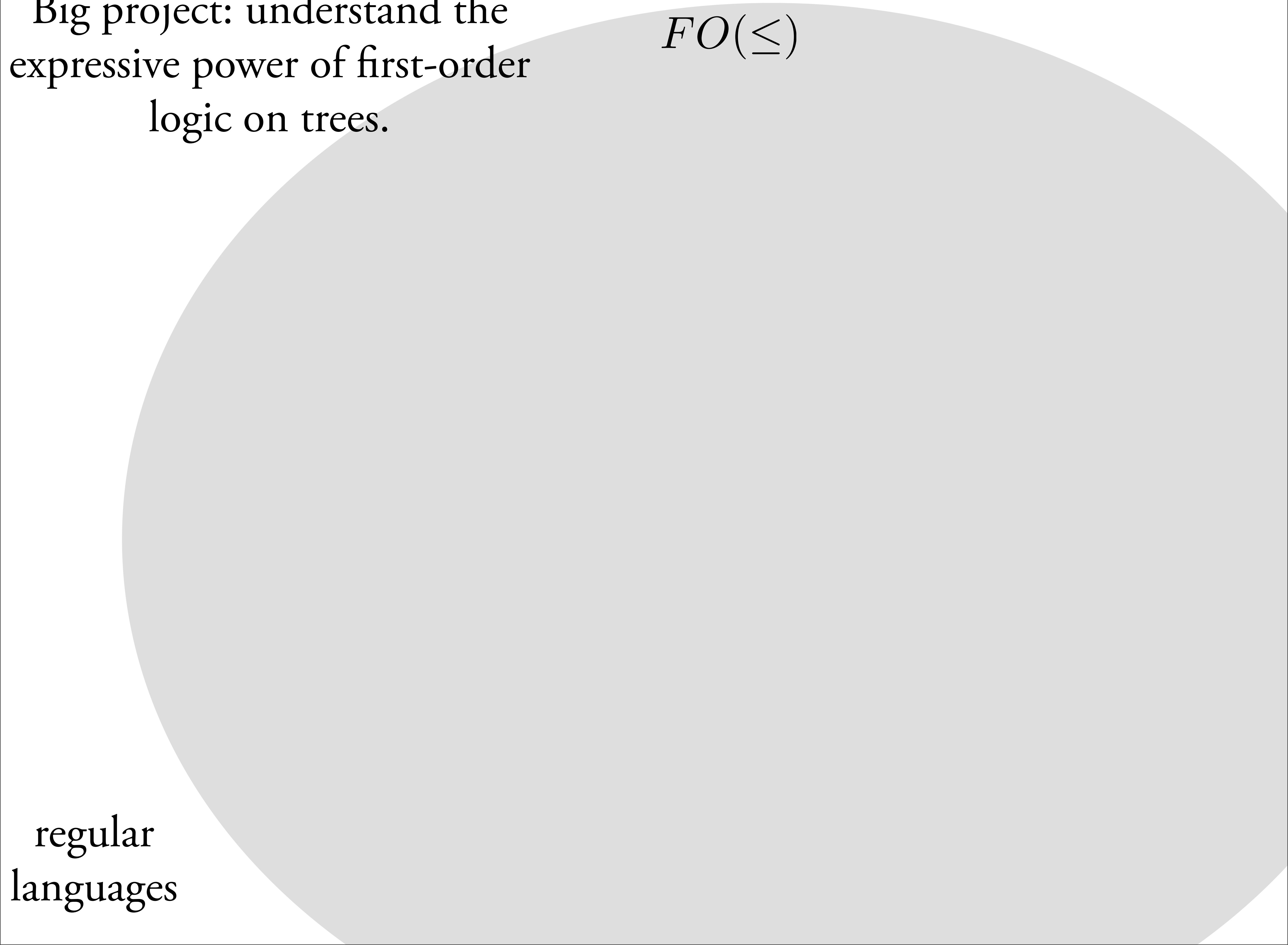


is confused with

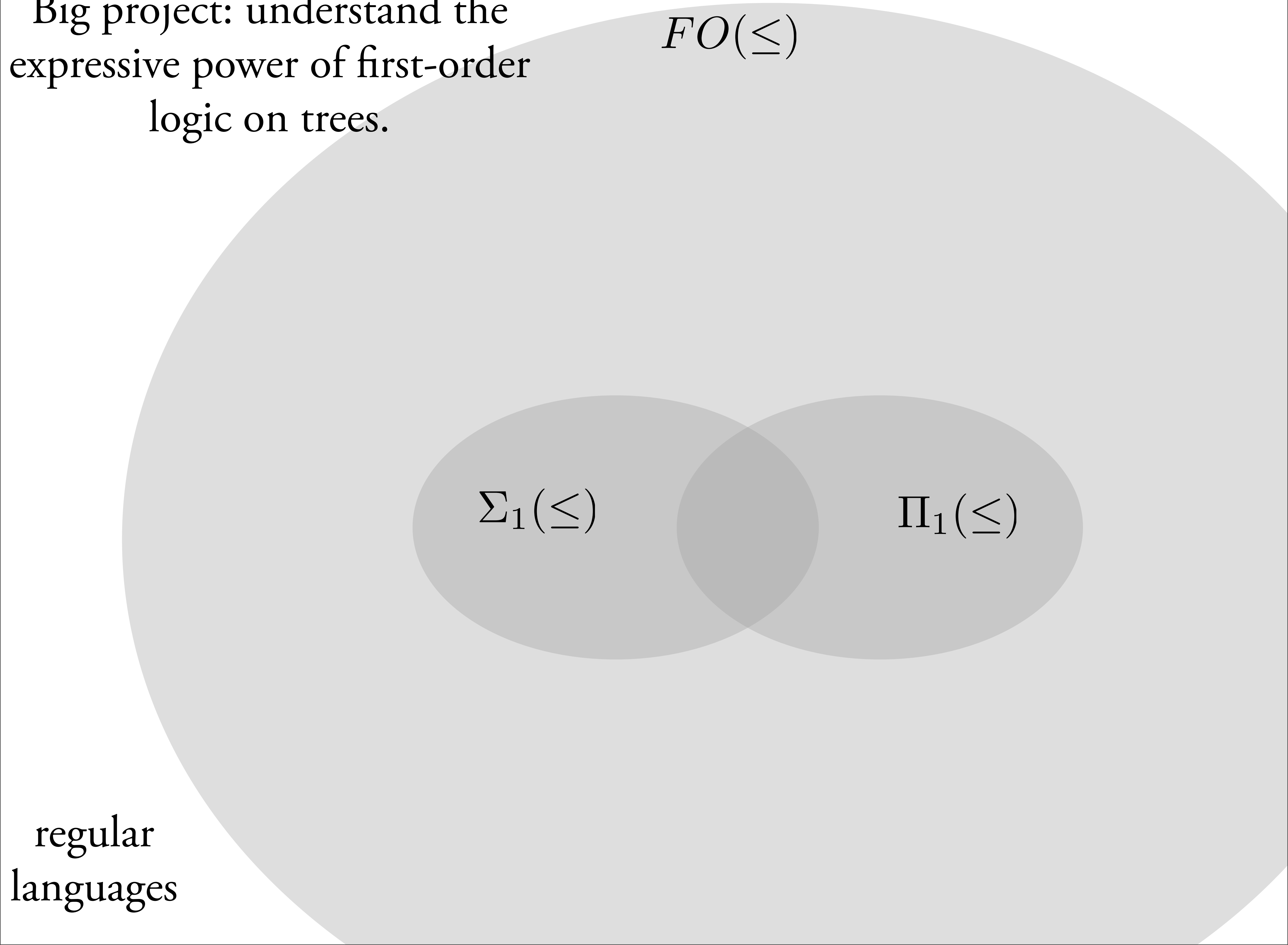


Big project: understand the expressive power of first-order logic on trees.

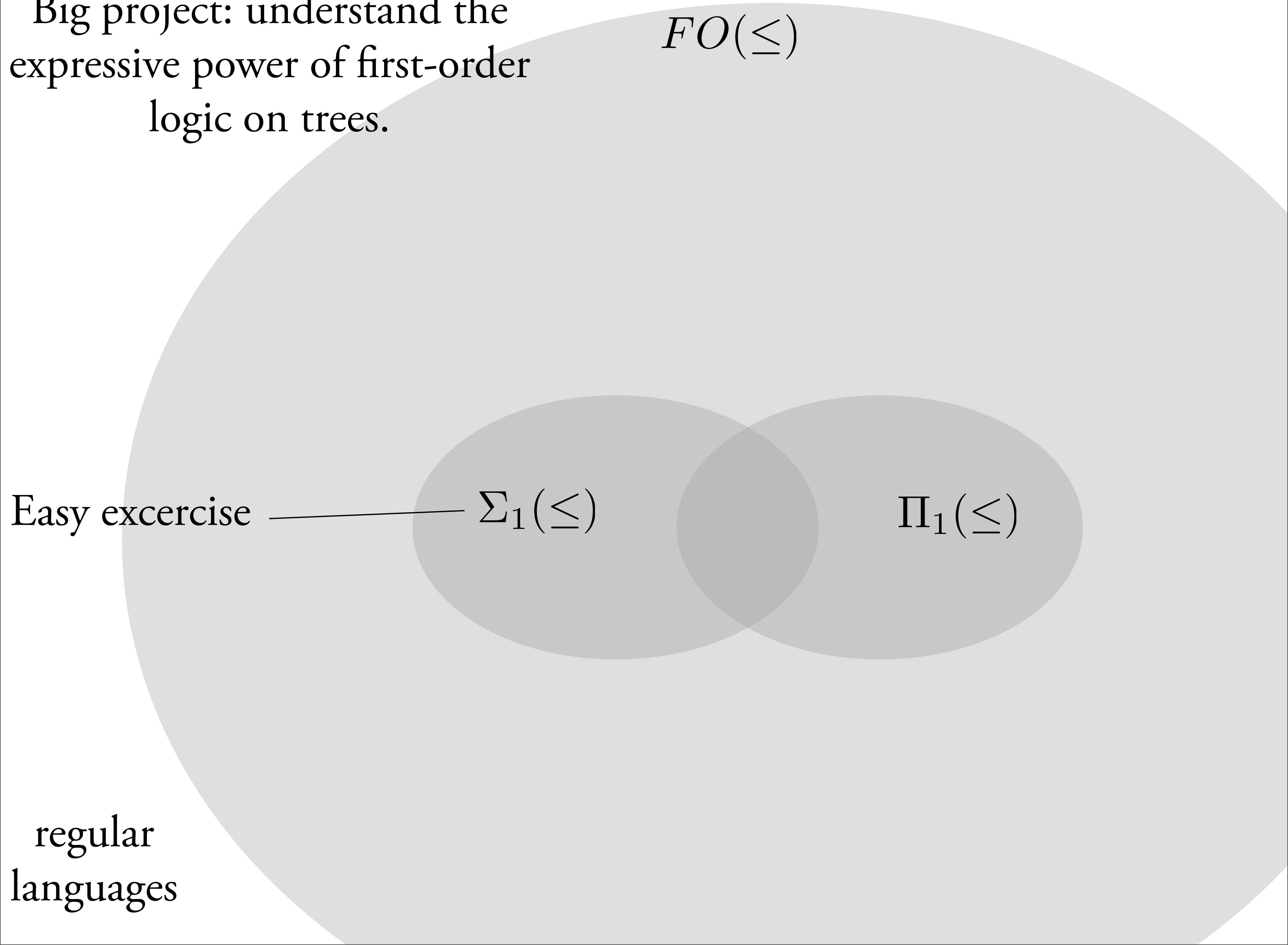
Big project: understand the expressive power of first-order logic on trees.



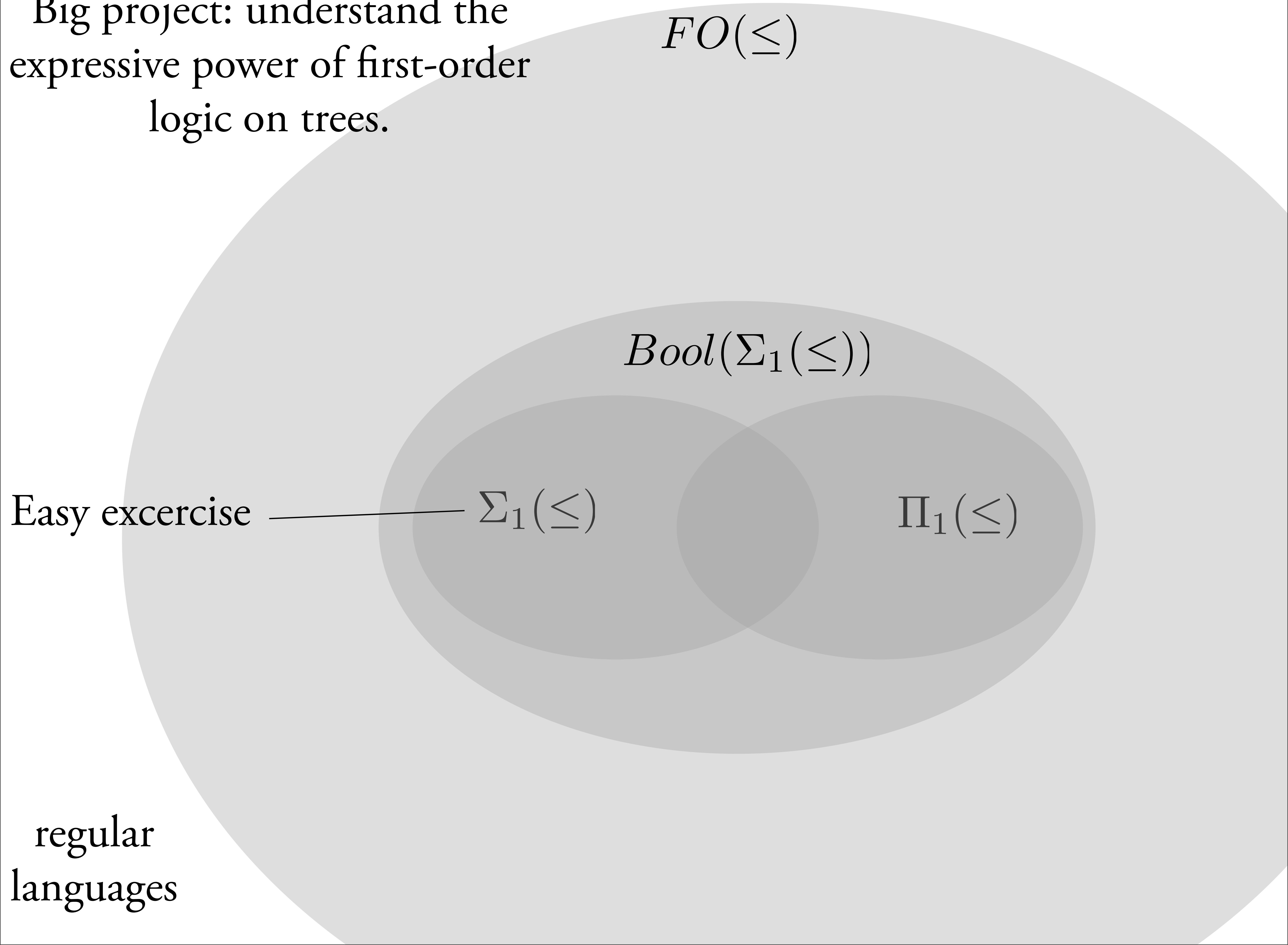
Big project: understand the expressive power of first-order logic on trees.



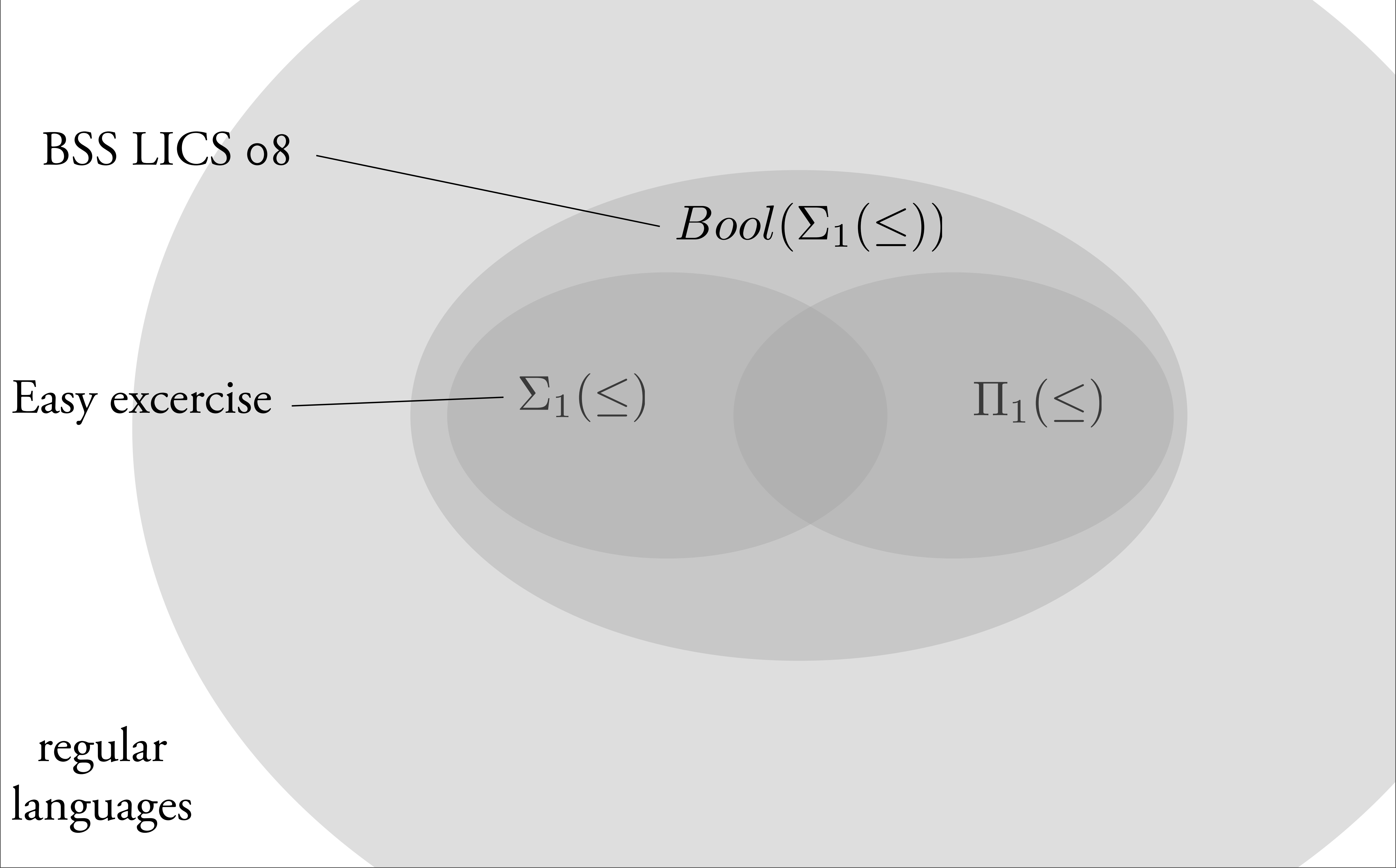
Big project: understand the expressive power of first-order logic on trees.



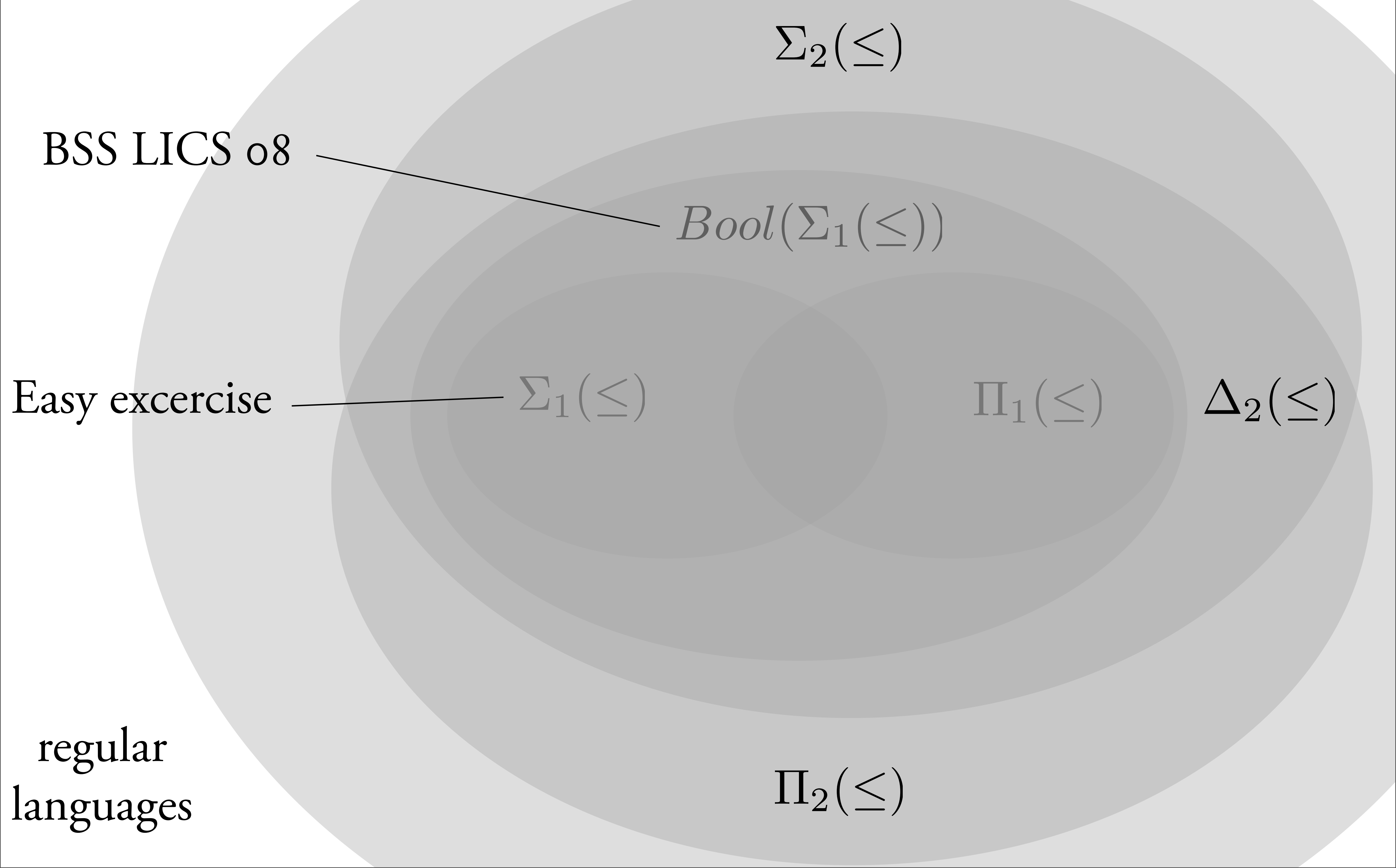
Big project: understand the expressive power of first-order logic on trees.

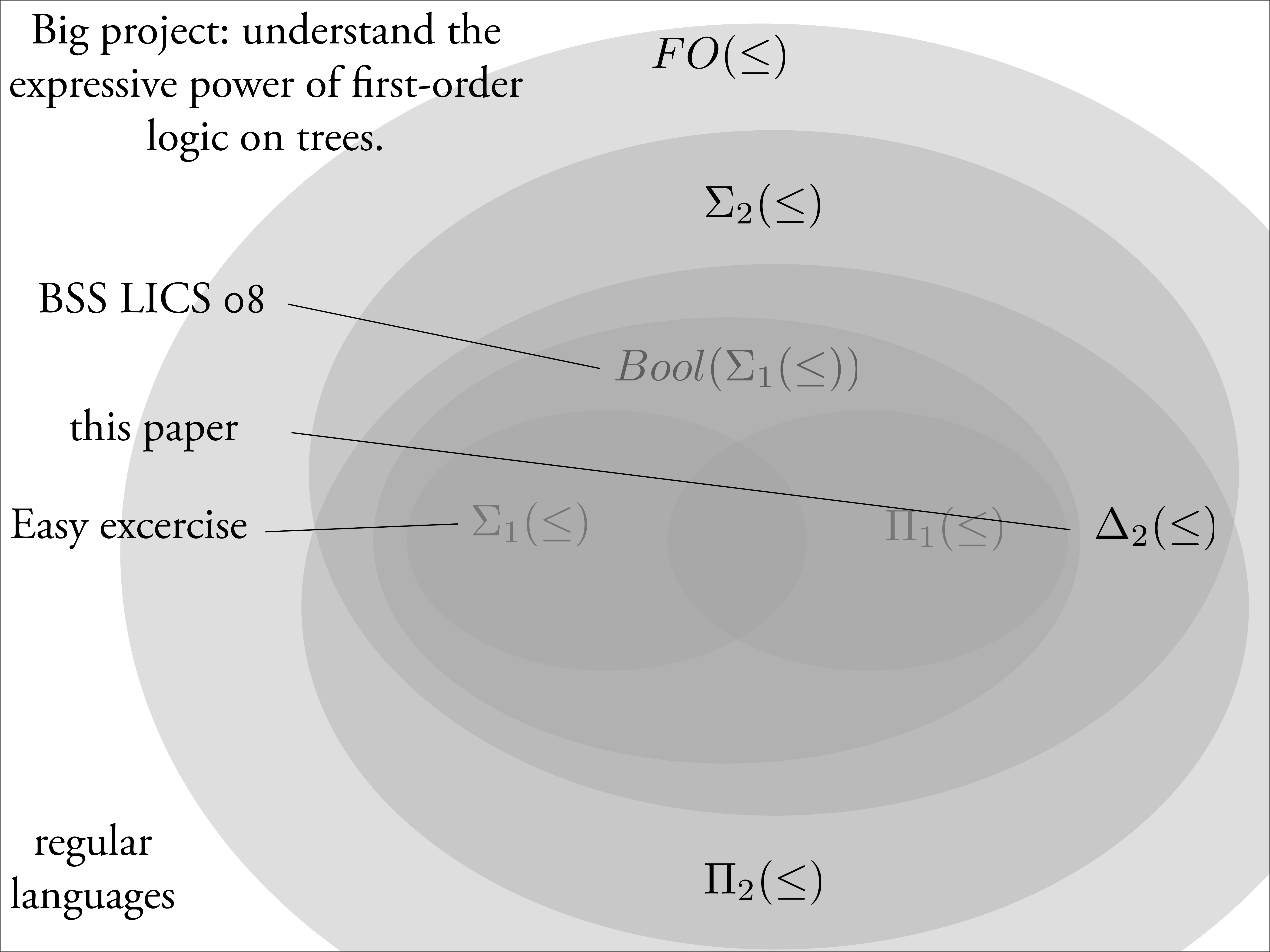


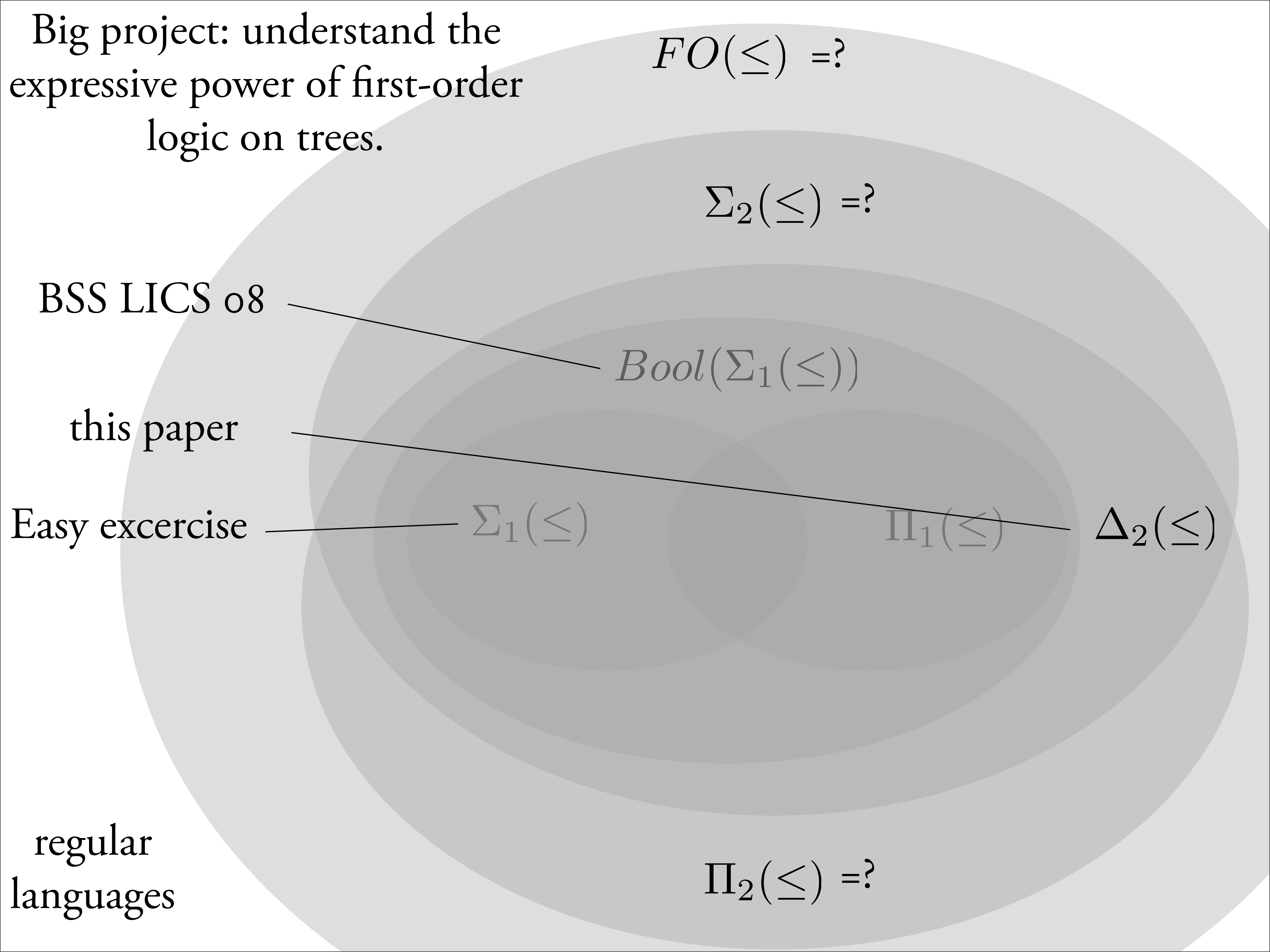
Big project: understand the expressive power of first-order logic on trees.



Big project: understand the expressive power of first-order logic on trees.




Big project: understand the expressive power of first-order logic on trees.
$FO(\leq)$
$\Sigma_2(\leq)$
BSS LICS 08
$Bool(\Sigma_1(\leq))$
this paper
Easy excercise
$\Sigma_1(\leq)$
$\Pi_1(\leq)$
$\Delta_2(\leq)$
regular languages
$\Pi_2(\leq)$



Big project: understand the expressive power of first-order logic on trees.
$FO(\leq)$ =?
$\Sigma_2(\leq)$ =?
BSS LICS 08
$Bool(\Sigma_1(\leq))$
this paper
Easy excercise
$\Sigma_1(\leq)$
$\Pi_1(\leq)$
$\Delta_2(\leq)$
regular languages
$\Pi_2(\leq)$ =?