Piecewise Testable Tree Languages

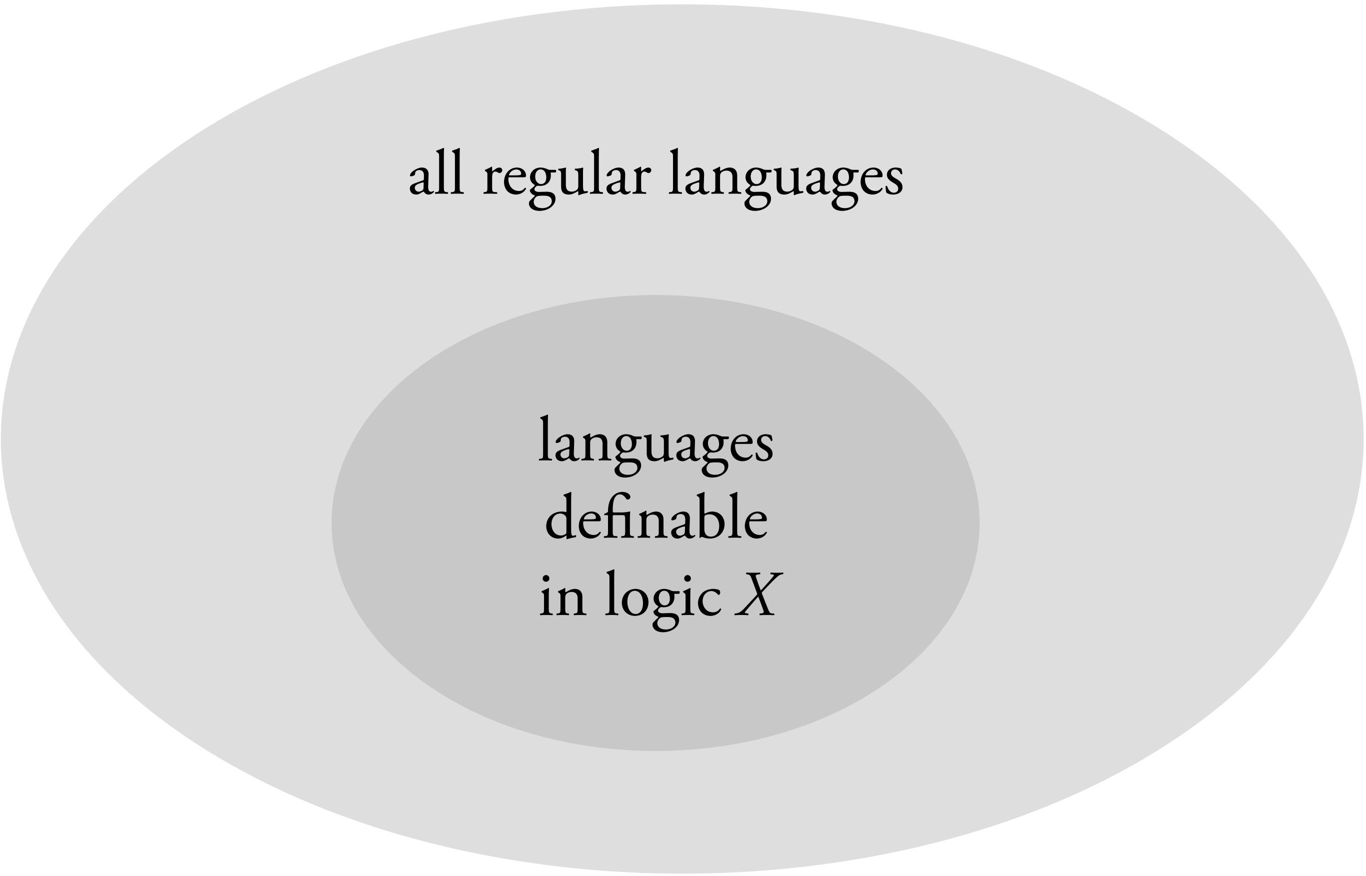
Mikołaj Bojańczyk, Luc Segoufin, Howard Straubing

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 =

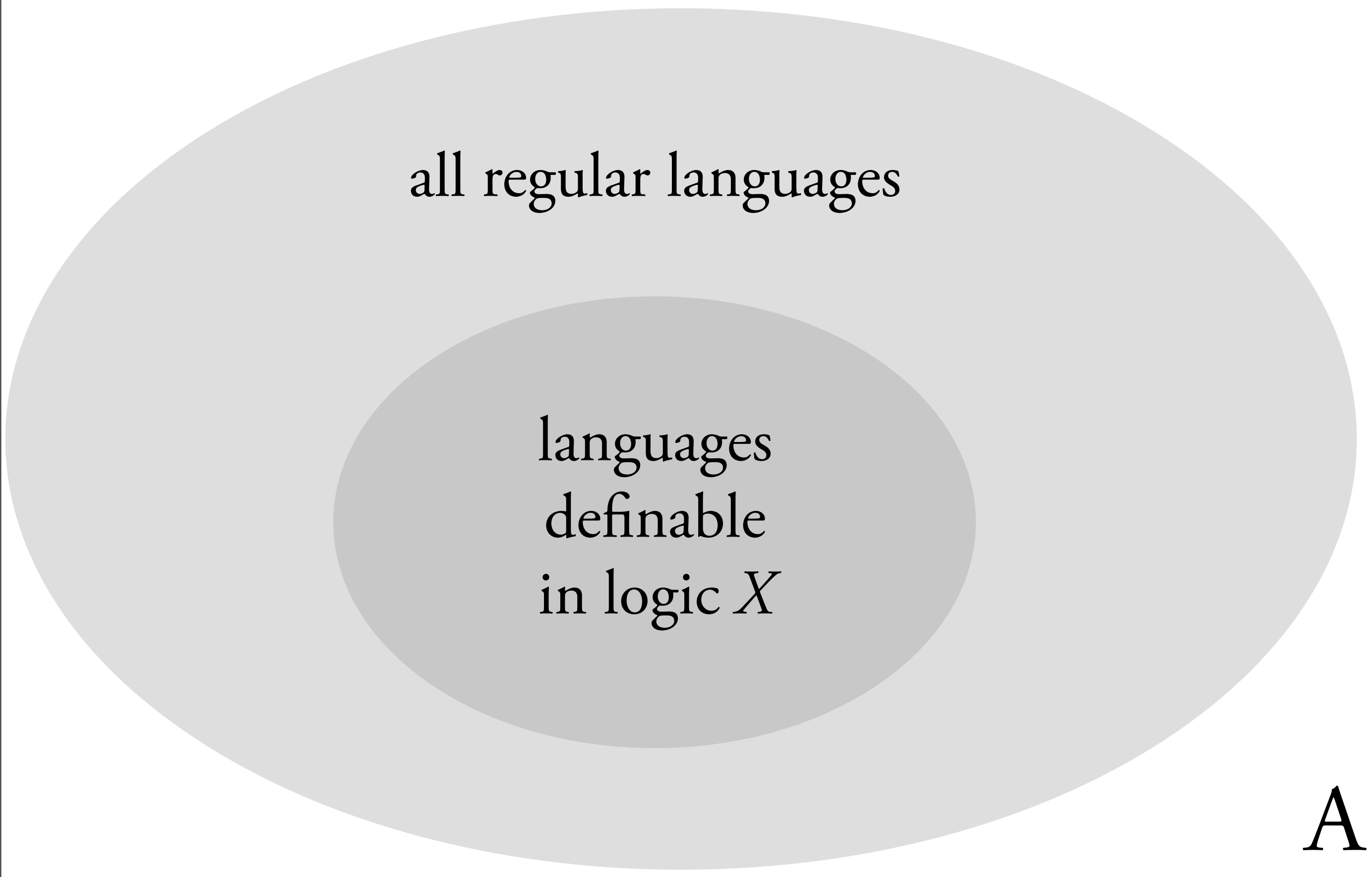
give na algorithm to decide if a language L is definable in X



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



Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is J-trivial.

a c b a c

a b is a piece of a c b a c

a b is a piece of a c b a c

Definition.
A word language is called *piecewise testable* if it is a boolean combination of languages “words that contain *w* as a piece”

a b is a piece of a c b a c

Definition.
A word language is called *piecewise testable* if it is a boolean combination of languages “words that contain w as a piece”

$\{ abc \}$ = contains piece *abc*, but no piece of length 4

a^*b^* = no piece *ba*

$a^*b^*a^*$ = no piece *bab*

a b is a piece of a c b a c

Definition.
A word language is called *piecewise testable* if it is a boolean combination of languages “words that contain *w* as a piece”

$\{ abc \}$ = contains piece *abc,* but no piece of length 4

a^*b^* = no piece *ba*

$a^*b^*a^*$ = no piece *bab*

Fact. A language is piecewise testable iff it can be defined by a boolean combination of $\Sigma_1(\leq)$ formulas.

$$\exists x \exists y\ a(x) \wedge b(y) \wedge x \leq y$$

Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is J-trivial.

Syntactic monoid of $L \subseteq \Sigma^*$

Syntactic monoid of $L \subseteq \Sigma^*$

Consider the two-sided Myhill-Nerode congruence

$$w \sim_L w'$$

holds if for every $u,v \in \Sigma^*$

$$uwv \in L \quad \text{iff} \quad uw'v \in L$$

Syntactic monoid of $L \subseteq \Sigma^*$

Consider the two-sided Myhill-Nerode congruence

$$w \sim_L w'$$

holds if for every $u,v \in \Sigma^*$

$$uwv \in L \quad \text{iff} \quad uw'v \in L$$

Elements of the syntactic monoid are equivalence classes of this congruence, the monoid operation is concatenation.

Syntactic monoid of $L \subseteq \Sigma^*$

Consider the two-sided Myhill-Nerode congruence

$$w \sim_L w'$$

holds if for every $u,v \in \Sigma^*$

$$uwv \in L \quad \text{iff} \quad uw'v \in L$$

Elements of the syntactic monoid are equivalence classes of this congruence, the monoid operation is concatenation.

Language	Its syntactic monoid		
$(aa)^*$	$(aa)^*$	$a(aa)^*$	
a^*ba^*	a^*	a^*ba^*	$a^*ba^*b(a+b)^*$

Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is J-trivial.

Infix relation in a monoid

For $s,t,u \in M$, we say s is an infix of tsu

We say $s,t \in M$ are in the same *J*-class if they are mutual infixes

Example. The syntactic monoid of $(aa)^*$ has two elements, $(aa)^*$ and $a(aa)^*$, which are in the same *J*-class.

A monoid is *J*-trivial if each *J*-class has one element.

Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is *J*-trivial.

Language	Its syntactic monoid		
$(aa)^*$	$(aa)^*$	$a(aa)^*$	
a^*ba^*	a^*	a^*ba^*	$a^*ba^*b(a+b)^*$
$a(a+b)^*$	ε	$a(a+b)^*$	$b(a+b)^*$

Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is *J*-trivial.

Language	Its syntactic monoid		
$(aa)^*$	$(aa)^*$	$a(aa)^*$	
a^*ba^*	a^*	a^*ba^*	$a^*ba^*b(a+b)^*$
$a(a+b)^*$	ε	$a(a+b)^*$	$b(a+b)^*$

Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is J-trivial.

Language	Its syntactic monoid			
$(aa)^*$	$(aa)^*$	$a(aa)^*$		✗
a^*ba^*	a^*	a^*ba^*	$a^*ba^*b(a+b)^*$	
$a(a+b)^*$	ε	$a(a+b)^*$	$b(a+b)^*$	

Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is *J*-trivial.

Language	Its syntactic monoid			
$(aa)^*$	$(aa)^*$	$a(aa)^*$		✗
a^*ba^*	a^*	a^*ba^*	$a^*ba^*b(a+b)^*$	
$a(a+b)^*$	ε	$a(a+b)^*$	$b(a+b)^*$	

Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is J-trivial.

Language	Its syntactic monoid			
$(aa)^*$	$(aa)^*$	$a(aa)^*$		✗
a^*ba^*	a^*	a^*ba^*	$a^*ba^*b(a+b)^*$	✓
$a(a+b)^*$	ε	$a(a+b)^*$	$b(a+b)^*$	

Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is *J*-trivial.

Language	Its syntactic monoid	
$(aa)^*$	[$(aa)^*$ $a(aa)^*$]	✗
a^*ba^*	[a^*] [a^*ba^*] [$a^*ba^*b(a+b)^*$]	✓
$a(a+b)^*$	[ε] [$a(a+b)^*$ $b(a+b)^*$]	

Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is *J*-trivial.

Language	Its syntactic monoid	
$(aa)^*$	[$(aa)^*$ $a(aa)^*$]	✗
a^*ba^*	[a^*] [a^*ba^*] [$a^*ba^*b(a+b)^*$]	✓
$a(a+b)^*$	[ε] [$a(a+b)^*$ $b(a+b)^*$]	✗

Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is J-trivial.

Language	Its syntactic monoid	
$(aa)^*$	$(aa)^*$ $a(aa)^*$	✗
a^*ba^*	a^*; a^*ba^*; $a^*ba^*b(a+b)^*$	✓
$a(a+b)^*$	ε; $a(a+b)^*$ $b(a+b)^*$	✗

Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is J-trivial.

Language	Its syntactic monoid			
$(aa)^*$	$(aa)^*$ $a(aa)^*$			✗
a^*ba^*	a^*	a^*ba^*	$a^*ba^*b(a+b)^*$	✓
$a(a+b)^*$	ε	$a(a+b)^*$ $b(a+b)^*$		✗

Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is J-trivial.

If s and t are in the same J-class, then for any n one can find representatives of s and t with the same pieces of size n.

w	uwv	$u'uwvv'$	$uu'uwvv'v$	$u'uu'uwvv'vv'v$	…
s	t	s	t	s	

Language	Its syntactic monoid	
$(aa)^*$	[$(aa)^*$ $a(aa)^*$]	✗
a^*ba^*	[a^*] [a^*ba^*] [$a^*ba^*b(a+b)^*$]	✓
$a(a+b)^*$	[ε] [$a(a+b)^*$ $b(a+b)^*$]	✗

Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is J-trivial.

Language	Its syntactic monoid			
$(aa)^*$	$(aa)^*$ $a(aa)^*$			✗
a^*ba^*	a^*	a^*ba^*	$a^*ba^*b(a+b)^*$	✓
$a(a+b)^*$	ε	$a(a+b)^*$ $b(a+b)^*$		✗

Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is J-trivial.

Language	Its syntactic monoid			
$(aa)^*$	$(aa)^*$ $a(aa)^*$			✗
a^*ba^*	a^*	a^*ba^*	$a^*ba^*b(a+b)^*$	✓
$a(a+b)^*$	ε	$a(a+b)^*$ $b(a+b)^*$		✗

Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is J-trivial.

Several arguments, all difficult.

What's the point of all this?

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

What's the point of all this?

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

What's the point of all this?

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

Theorem. (Schützenberger, Thérien / Wilke)
The following are equivalent for a word language:
– L is definable in two-variable first-order logic
– L can be defined by a type of unambiguous expression
– the syntactic monoid of L is in DA

What's the point of all this?

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

Theorem. (Schützenberger, Thérien / Wilke)
The following are equivalent for a word language:
– L is definable in two-variable first-order logic
– L can be defined by a type of unambiguous expression
– the syntactic monoid of L is in DA

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

What's the point of all this?

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

What about trees?

Theorem. (Schützenberger, Thérien / Wilke)
The following are equivalent for a word language:
– L is definable in two-variable first-order logic
– L can be defined by a type of unambiguous expression
– the syntactic monoid of L is in DA

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

What's the point of all this?

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

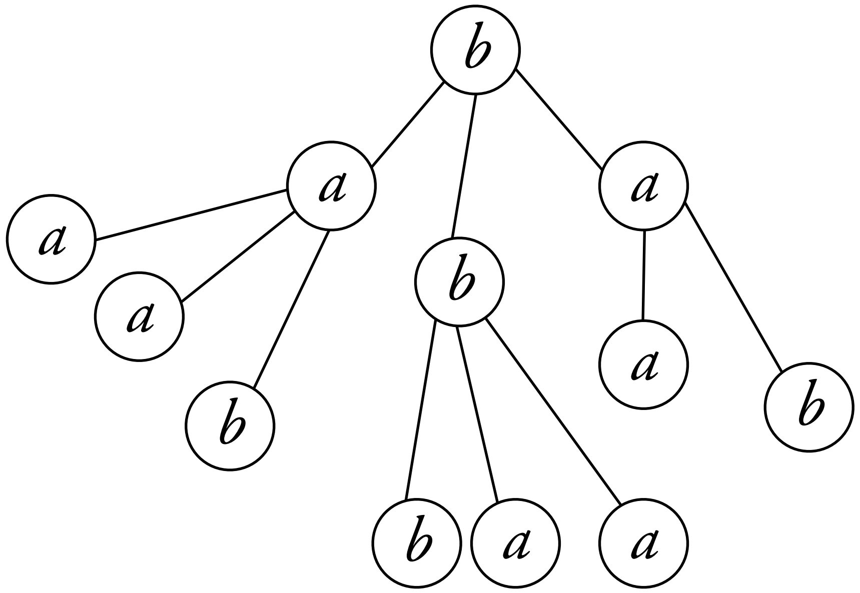
Theorem. (Schützenberger, Thérien / Wilke)
The following are equivalent for a word language:
– L is definable in two-variable first-order logic
– L can be defined by a type of unambiguous expression
– the syntactic monoid of L is in DA

What about trees?

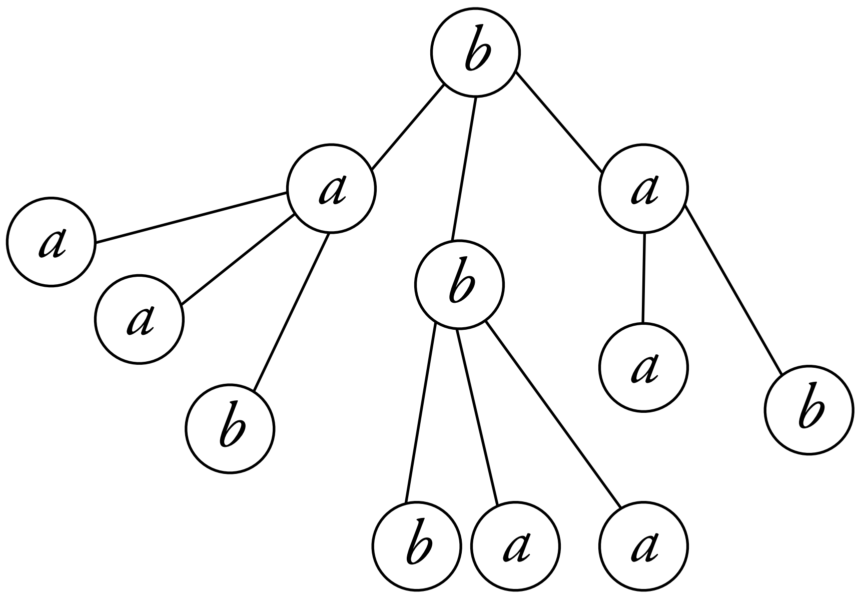
This paper is part of a program to extend the algebra-logic connection to trees

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

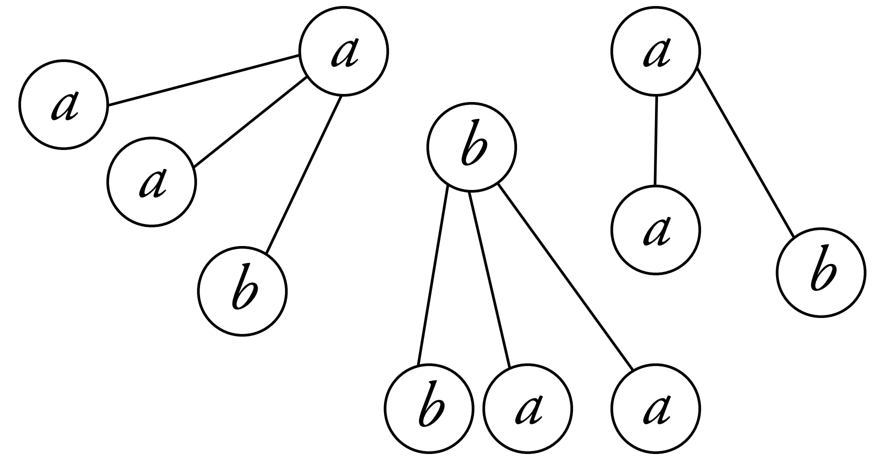
A *tree* is finite, unranked and labeled



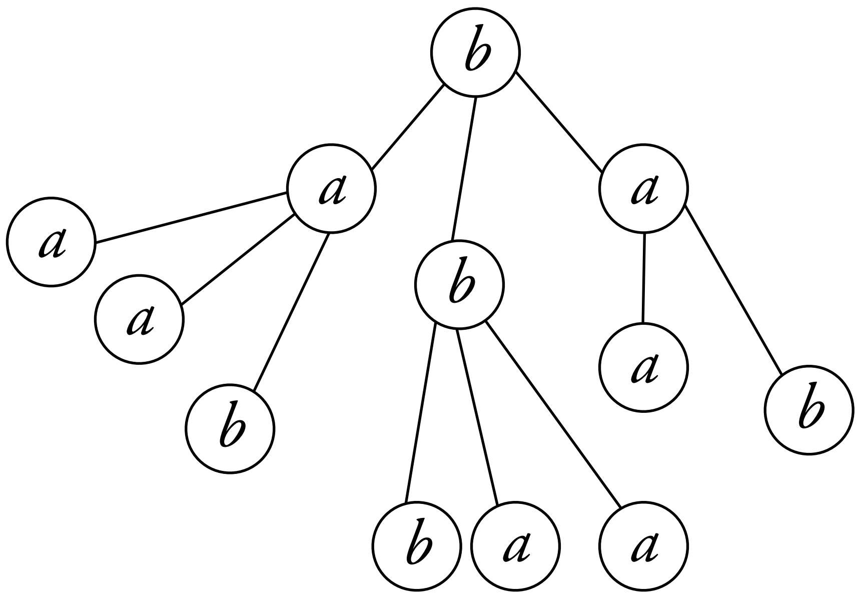
A *tree* is finite, unranked and labeled



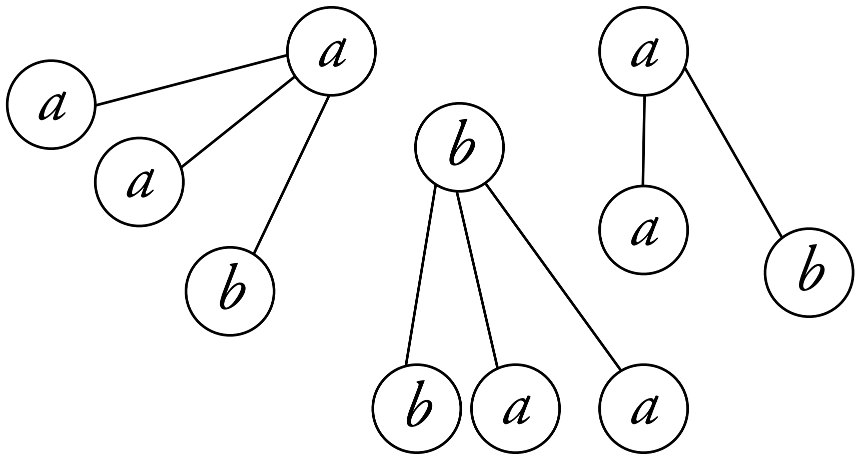
A *forest* is a sequence of trees



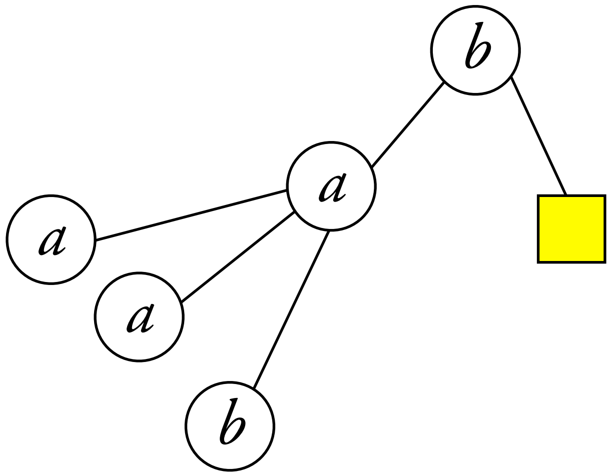
A *tree* is finite, unranked and labeled



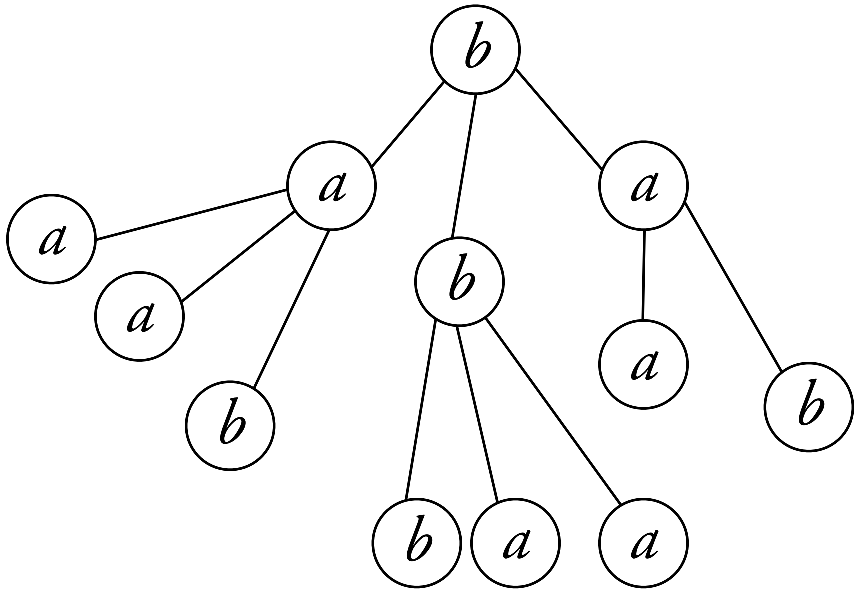
A *forest* is a sequence of trees



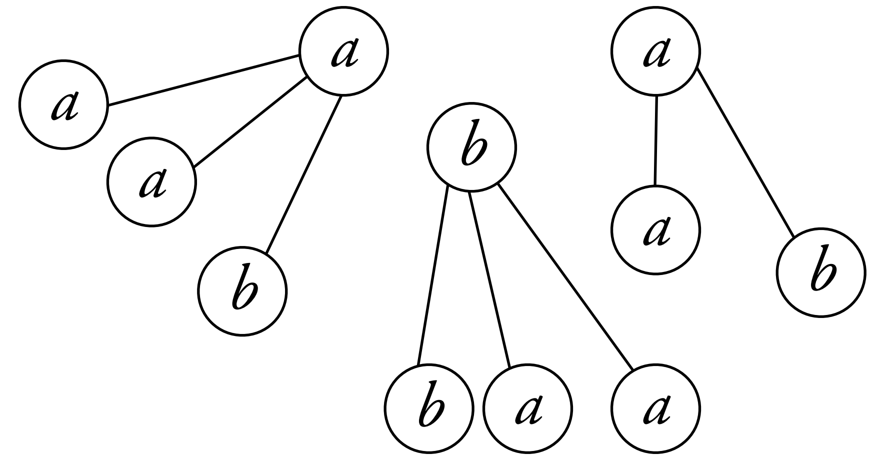
A *context* is a forest with a hole in a leaf



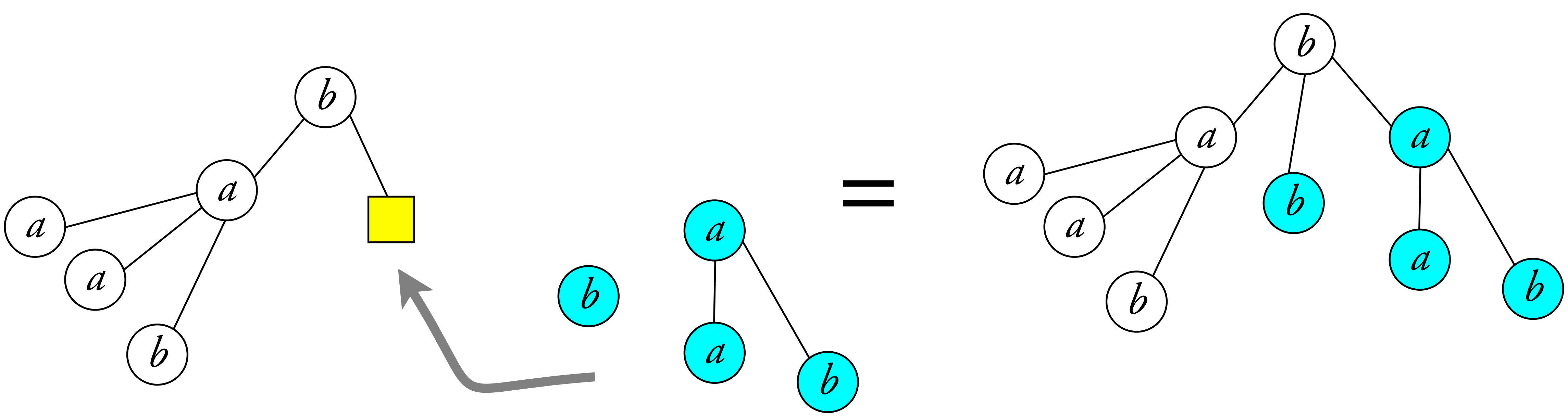
A *tree* is finite, unranked and labeled



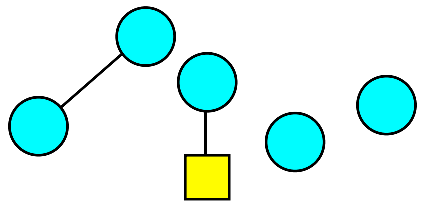
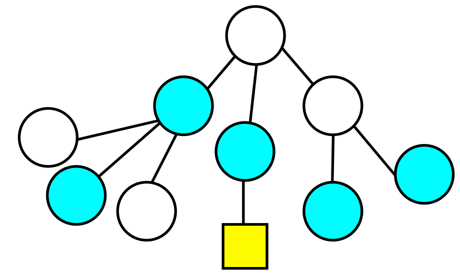
A *forest* is a sequence of trees

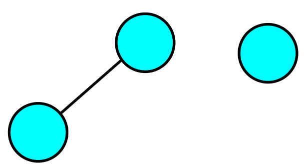
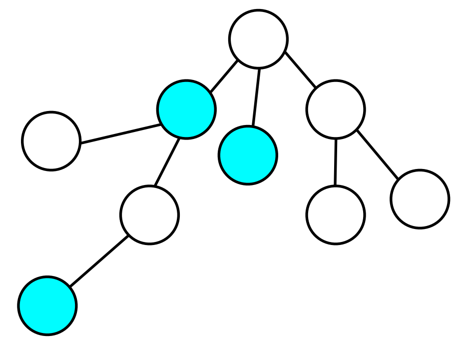


A *context* is a forest with a hole in a leaf

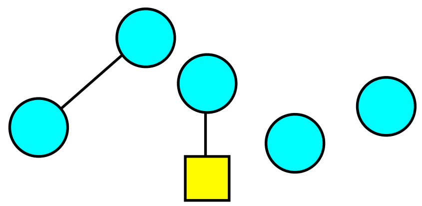
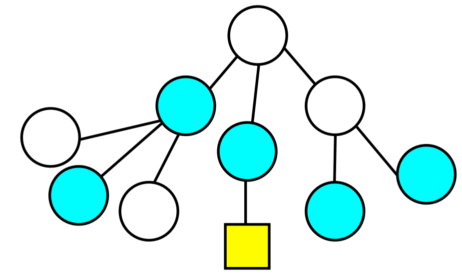


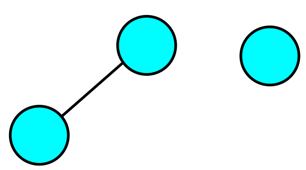
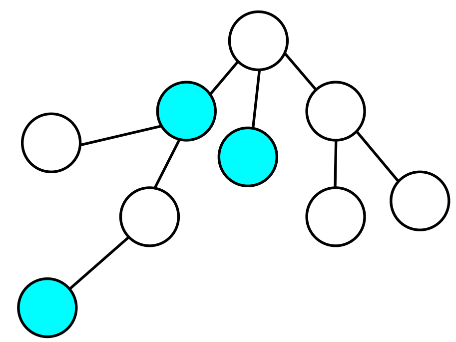
Notion of piece for forests and contexts.

 is a piece of 

 is a piece of 

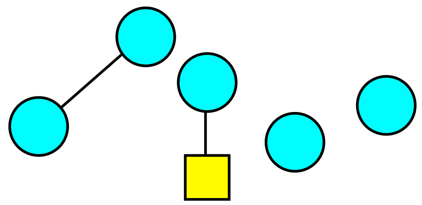
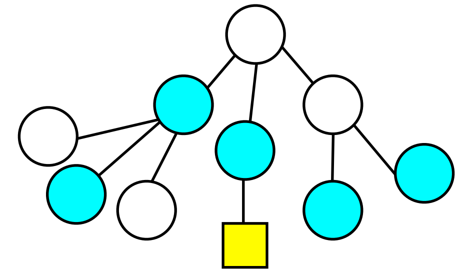
Notion of piece for forests and contexts.

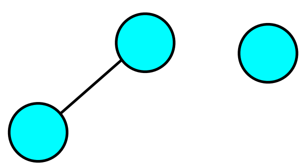
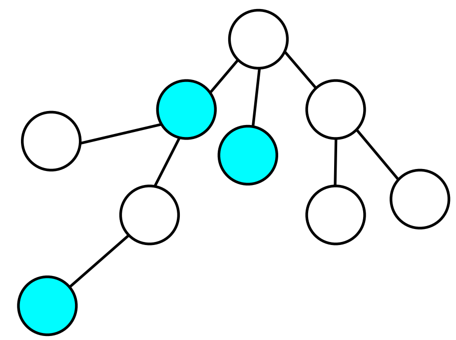
 is a piece of 

 is a piece of 

Definition.
A forest language is called *piecewise testable* if it is a boolean combination of languages "forests that contain t as a piece"

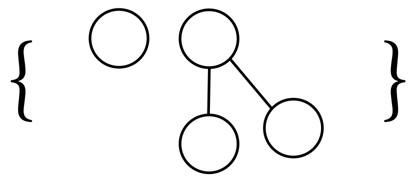
Notion of piece for forests and contexts.

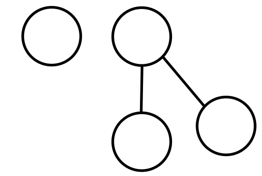
 is a piece of 

 is a piece of 

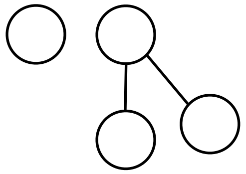
Definition.
A forest language is called *piecewise testable* if it is a boolean combination of languages “forests that contain *t* as a piece”

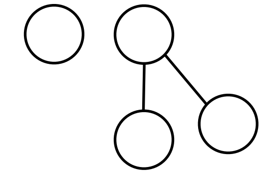
Fact. A forest language is piecewise testable iff it can be defined by a boolean combination of $\Sigma_1(\leq, \leq_{lex})$ formulas.



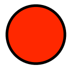
contains piece 

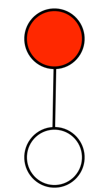
contains no piece with 5 nodes

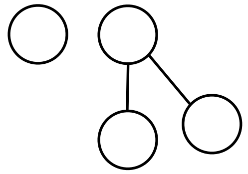
{  }

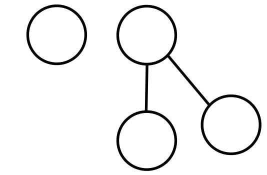
contains piece 

contains no piece with 5 nodes

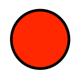
all leaves are 

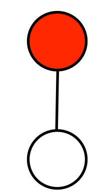
contains no piece 

{  }

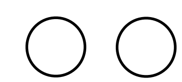
contains piece 

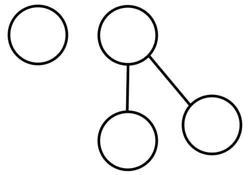
contains no piece with 5 nodes

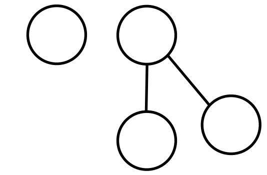
all leaves are 

contains no piece 

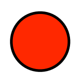
forest is a word (vertically)

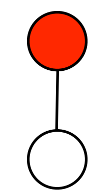
contains no piece 

{  }

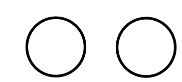
contains piece 

contains no piece with 5 nodes

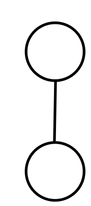
all leaves are 

contains no piece 

forest is a word (vertically)

contains no piece 

forest is a word (horizontally)

contains no piece 

We want the forest extension of:

Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is *J*-trivial.

We want the forest extension of:

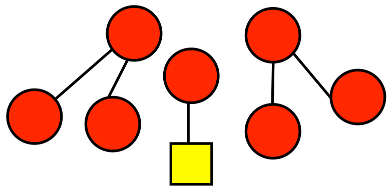
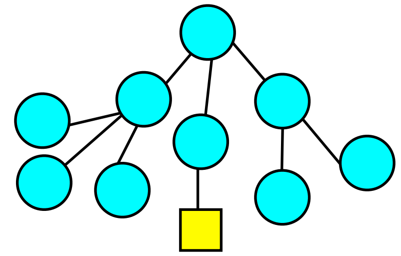
Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is J-trivial.

What is a syntactic monoid for forest languages?

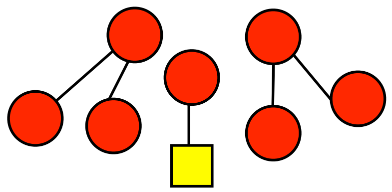
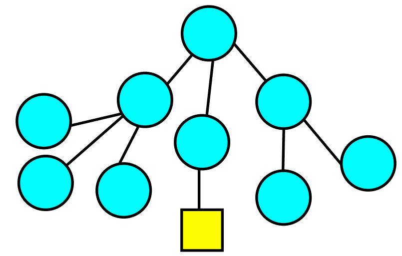
Although a definition exists (forest algebra), here we will only talk about Myhill-Nerode equivalence.

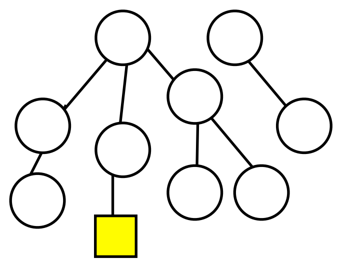
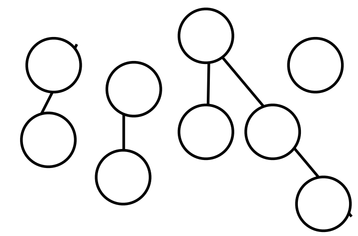
Myhill-Nerode congruence for a forest language L.

Myhill-Nerode congruence for a forest language *L*.

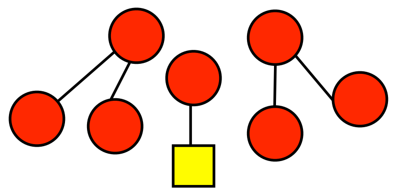
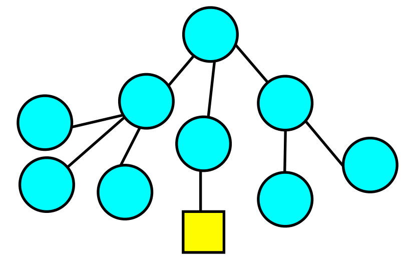
Two contexts  and  are called *L*-equivalent if

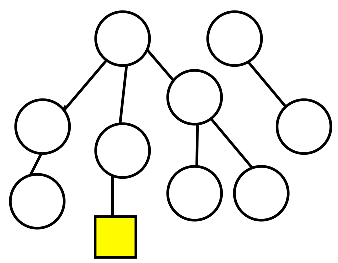
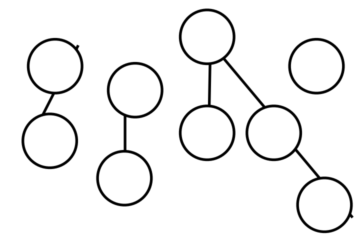
Myhill-Nerode congruence for a forest language L.

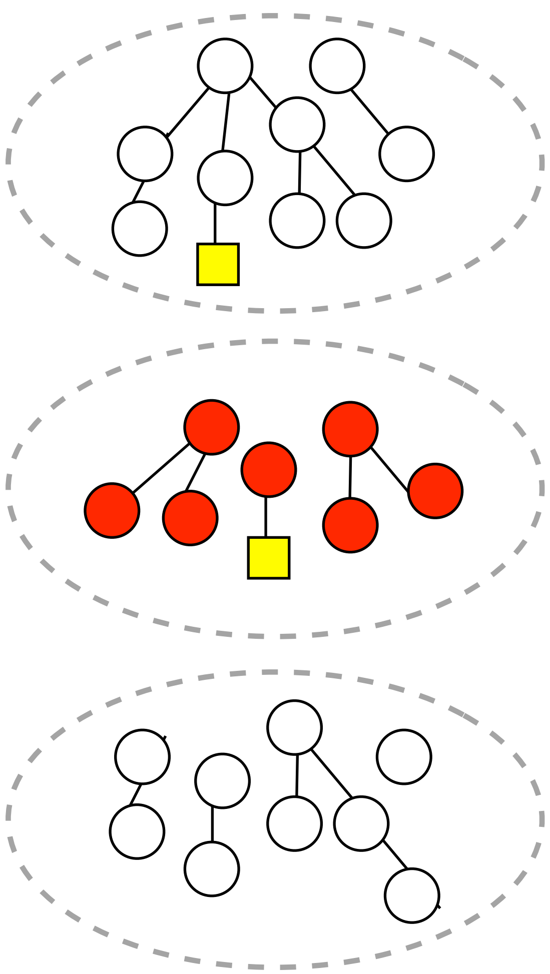
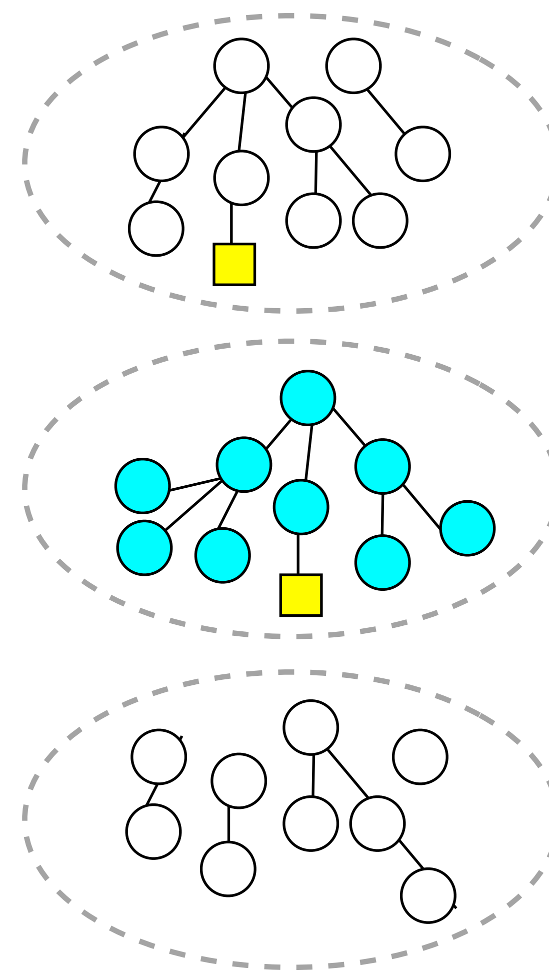
Two contexts  and  are called L-equivalent if

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  $\in L$

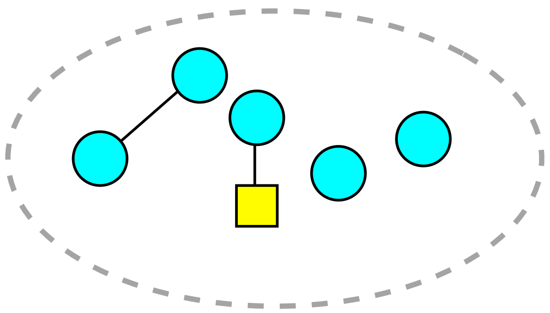
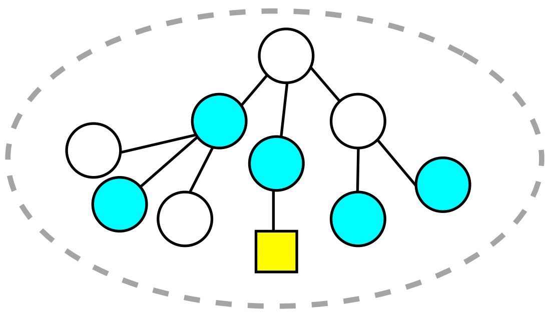
Main Theorem.

A forest language is piecewise testable iff

the following holds for all sufficiently large n

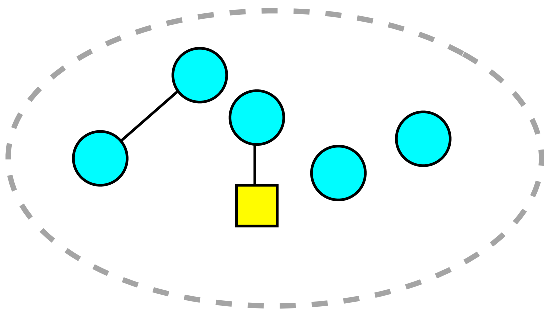
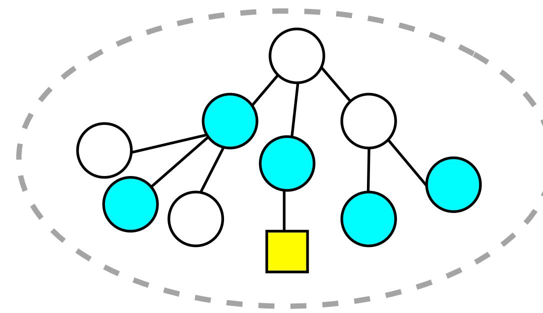
Main Theorem.

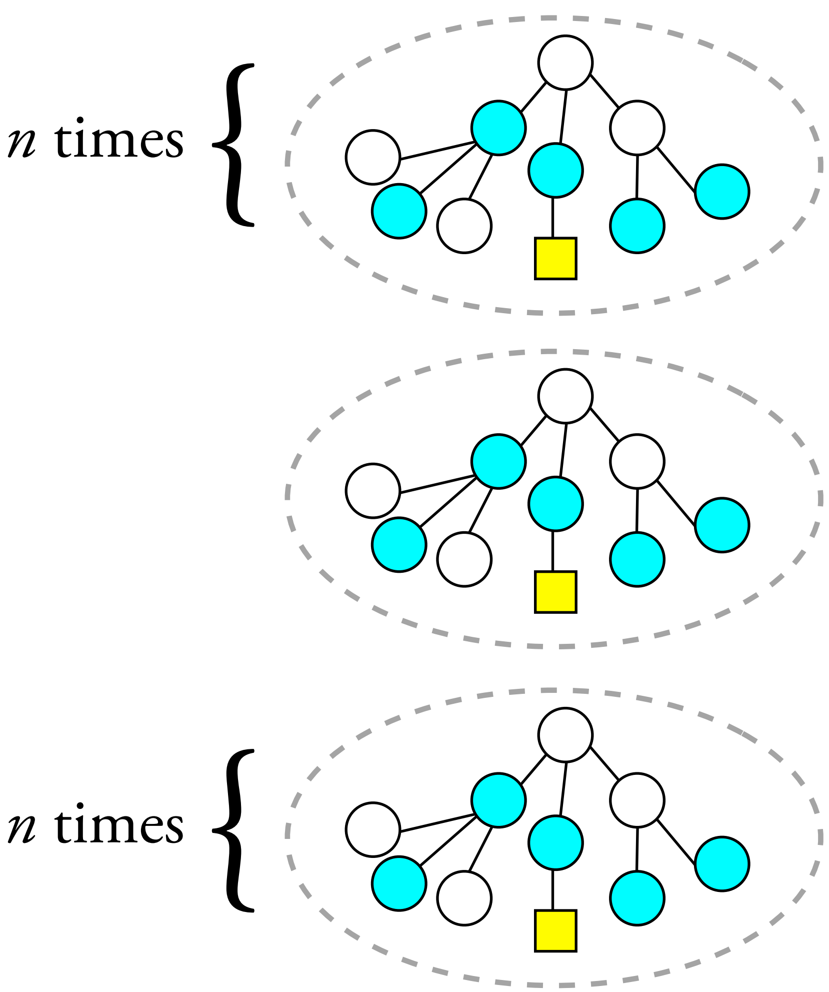
A forest language is piecewise testable iff
the following holds for all sufficiently large n

if  is a piece of  , then

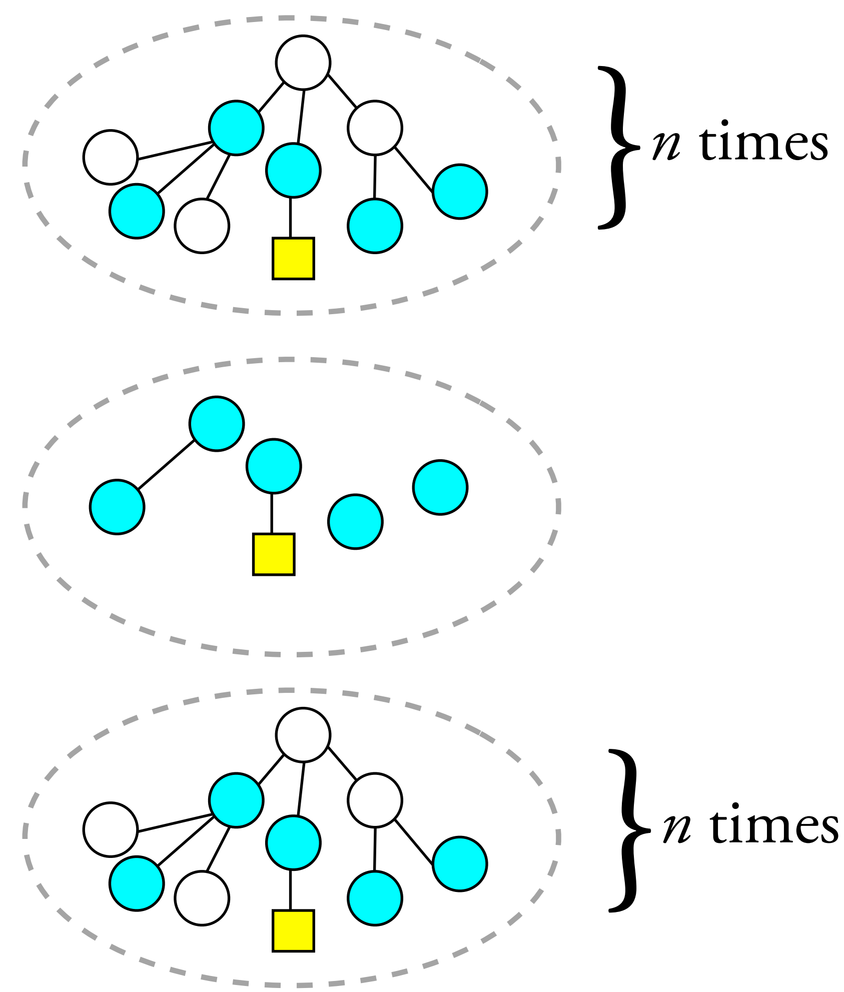
Main Theorem.

A forest language is piecewise testable iff the following holds for all sufficiently large n

if  is a piece of  , then



is equivalent to



Main Theorem.

A forest language is piecewise testable iff
the following holds for all sufficiently large n

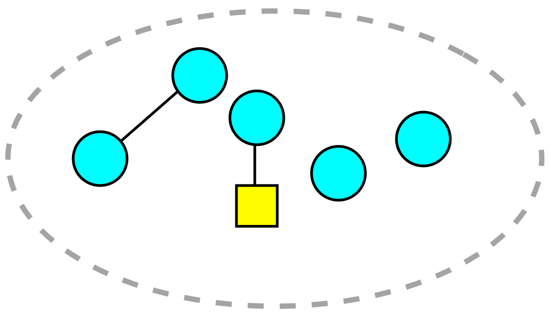
This criterion is decidable.

We also have variants of the theorem for

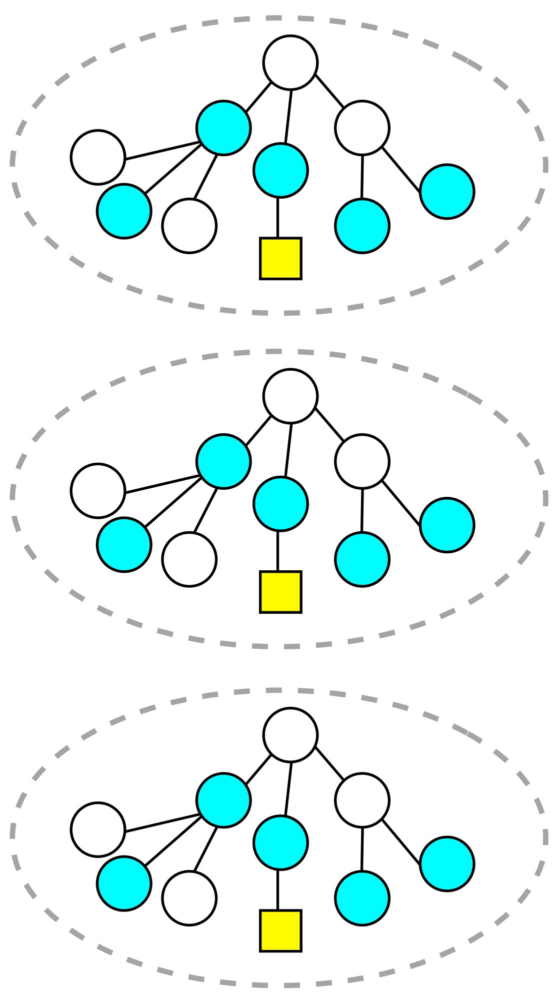
– tree languages

– commutative pieces

– pieces with closest common ancestor

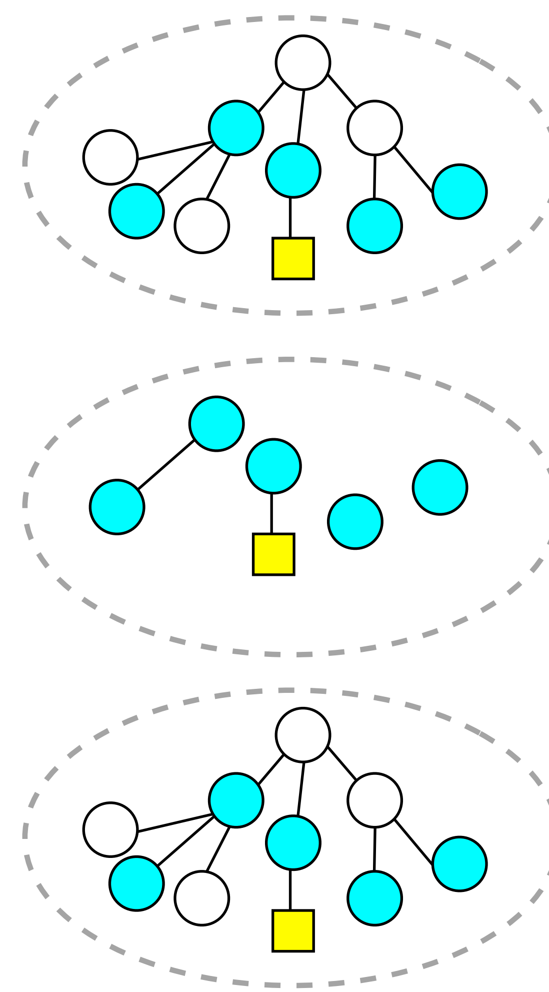
if  is a piece of …, then

n times



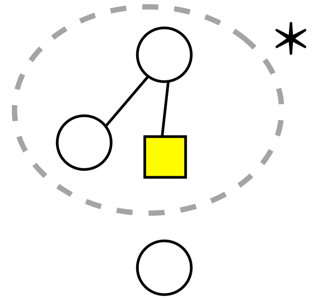
n times

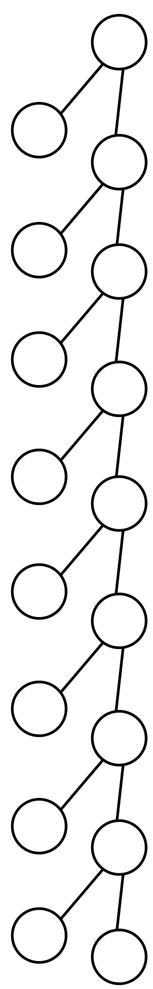
is equivalent to



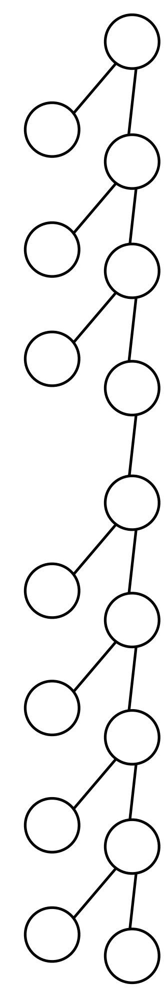
n times

n times

The language  has a J-trivial syntactic monoid, but is not piecewise testable

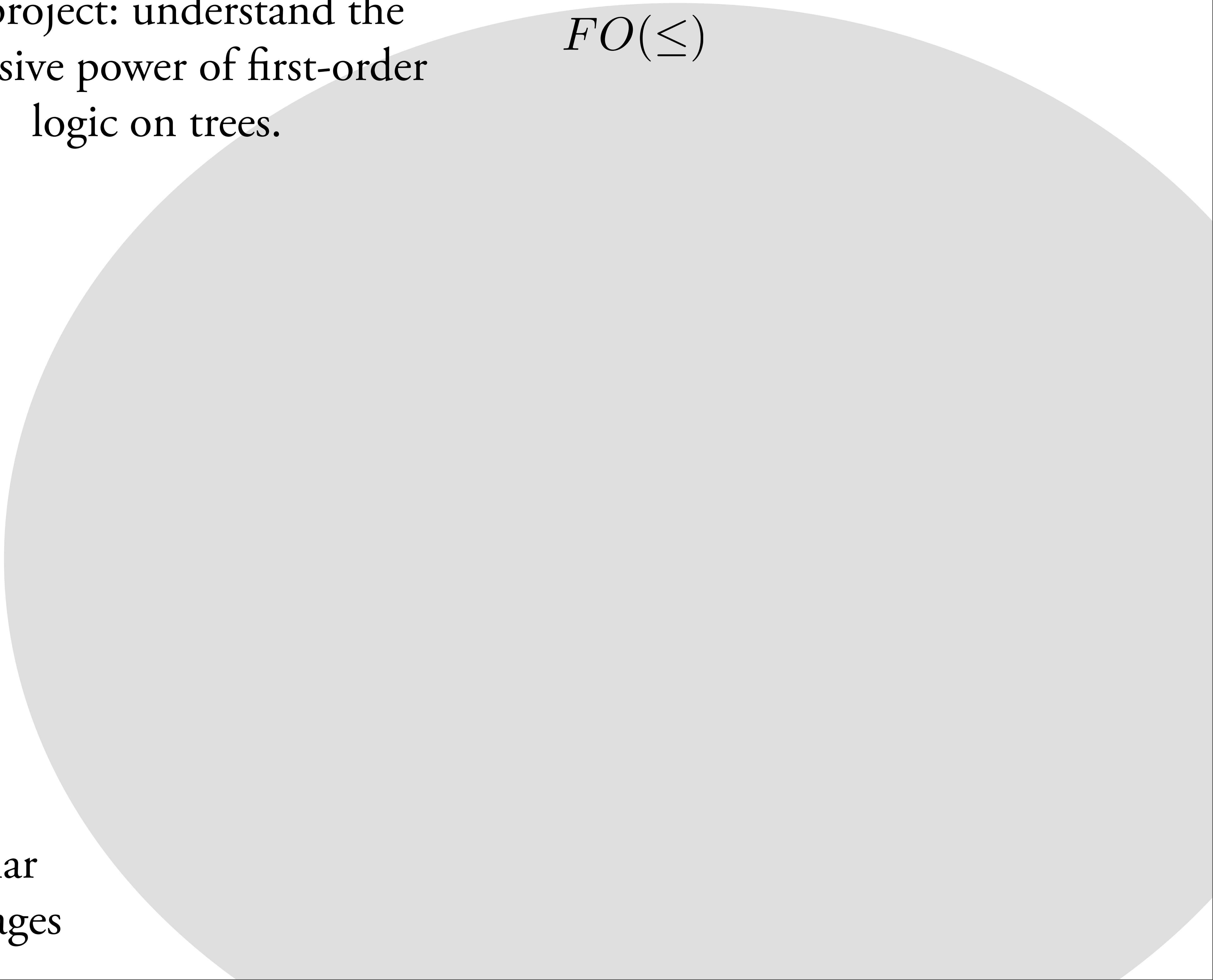


is confused with



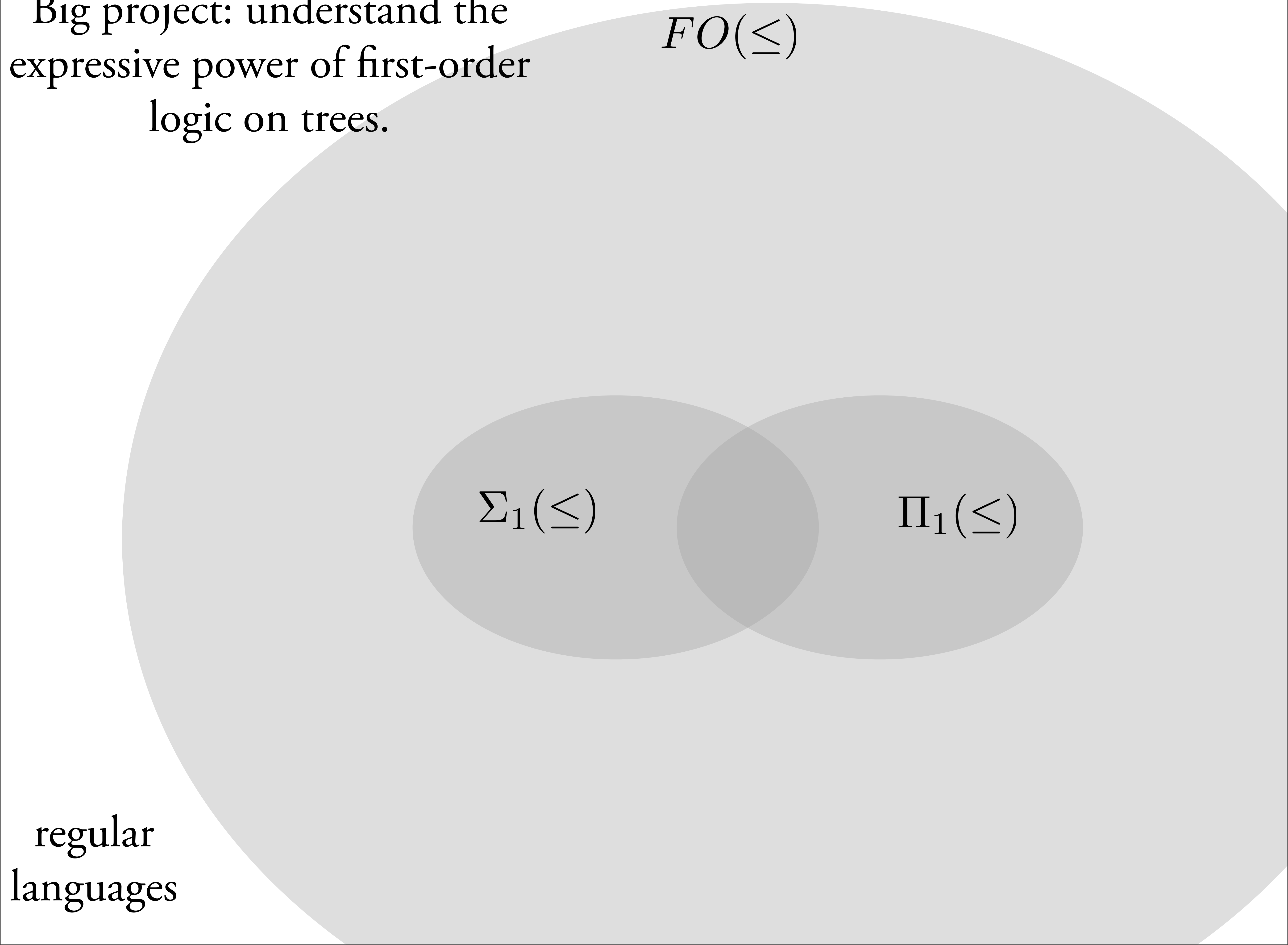
Big project: understand the expressive power of first-order logic on trees.

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

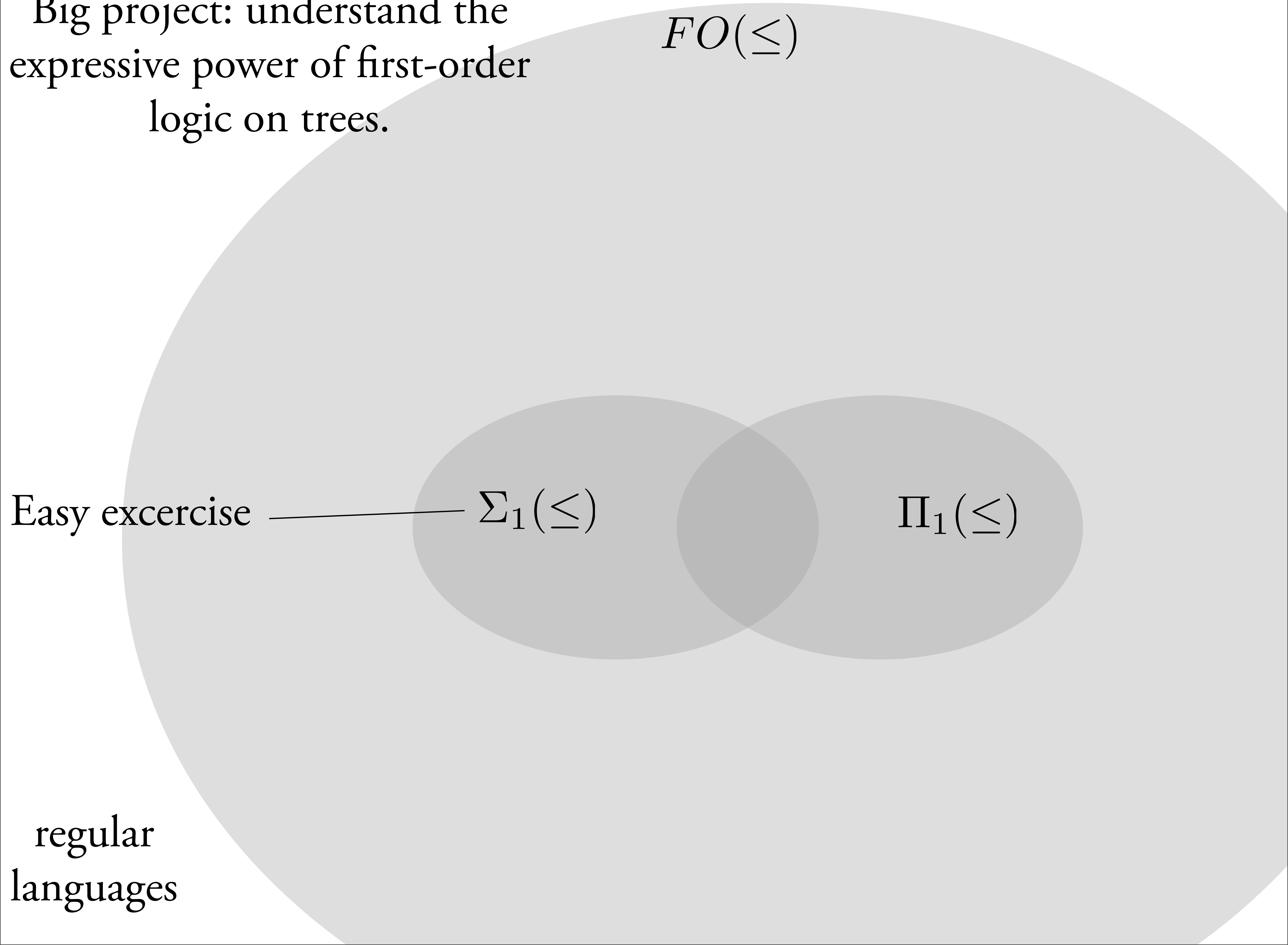


regular languages

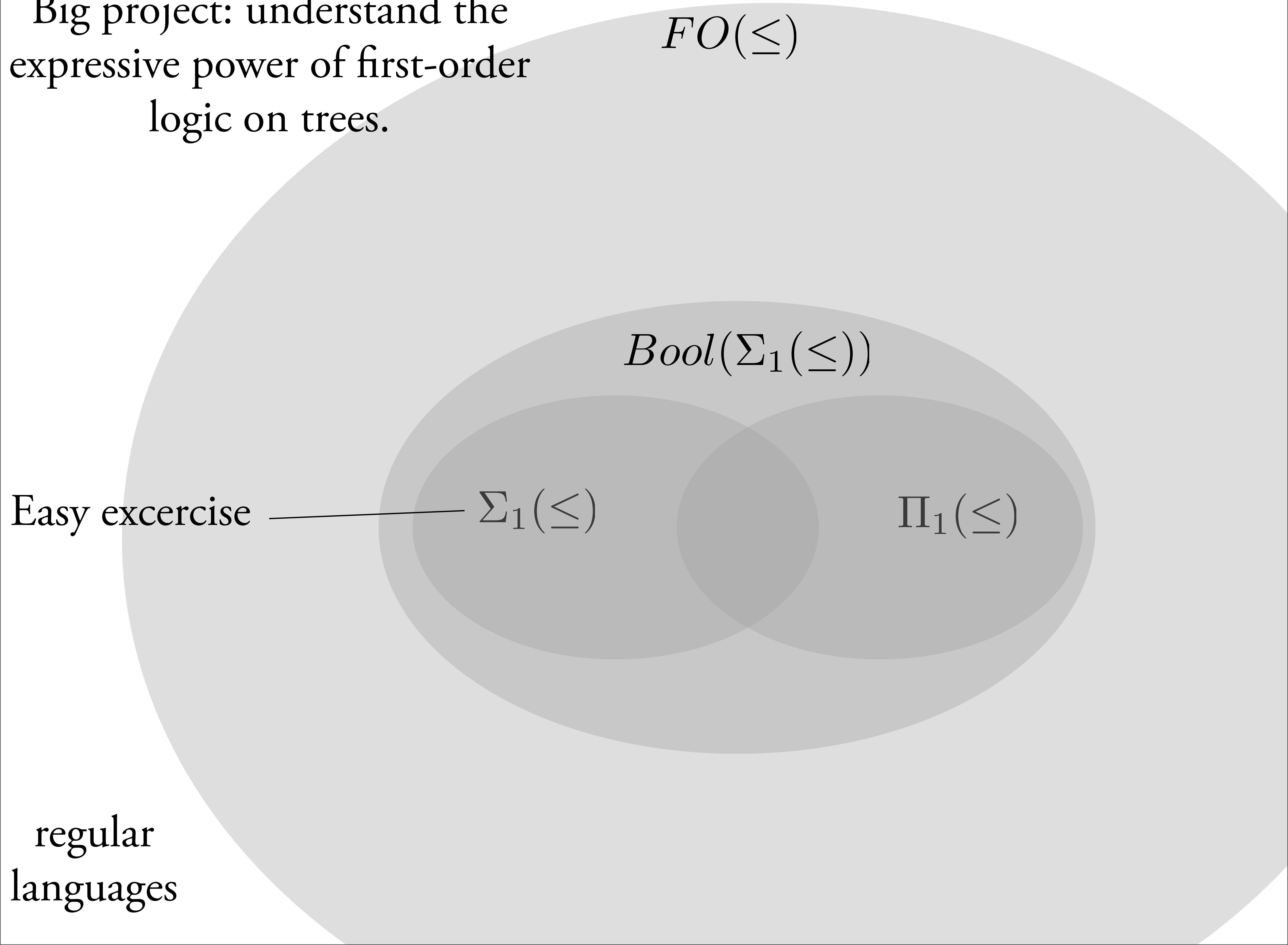
Big project: understand the expressive power of first-order logic on trees.

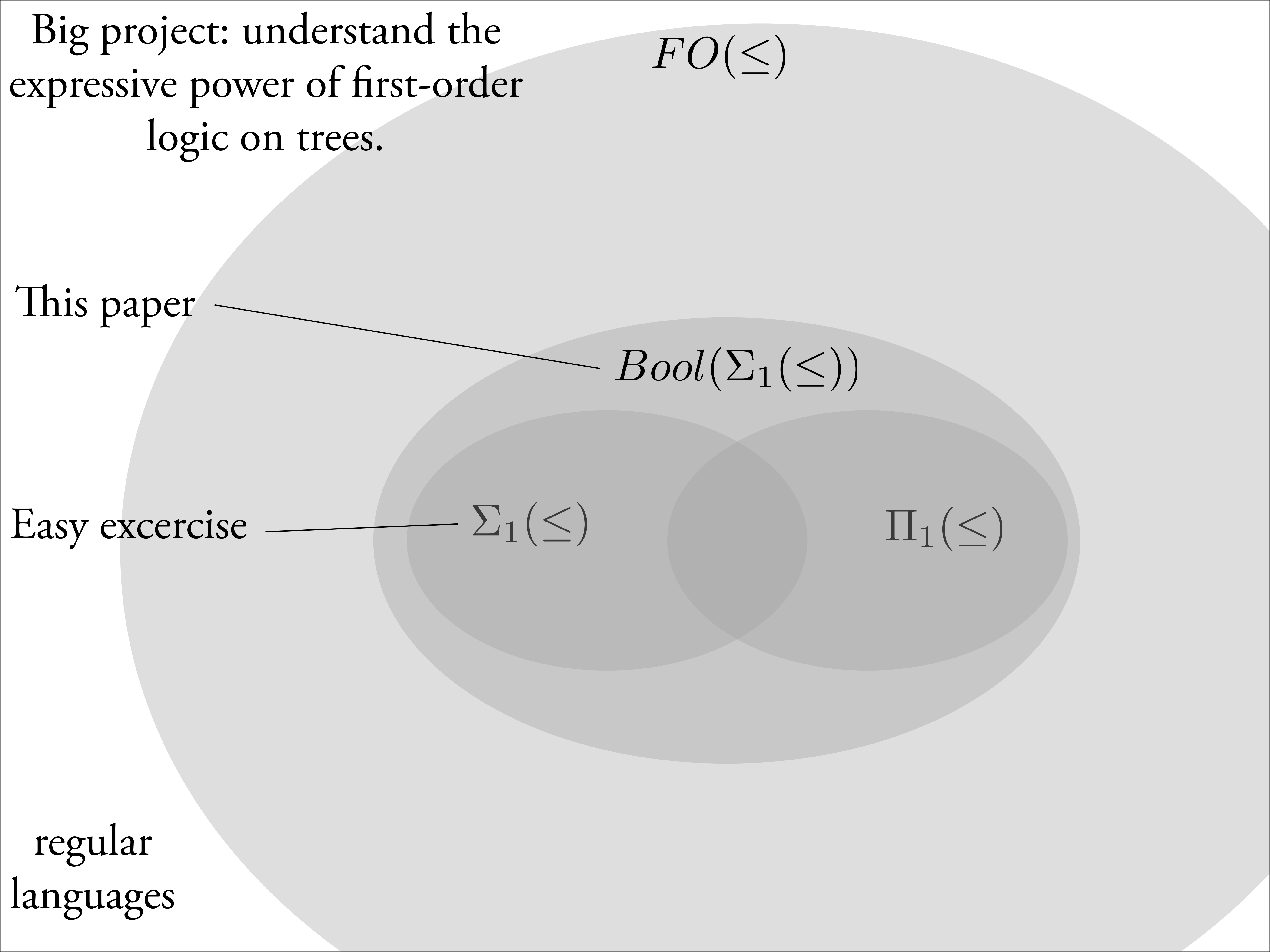


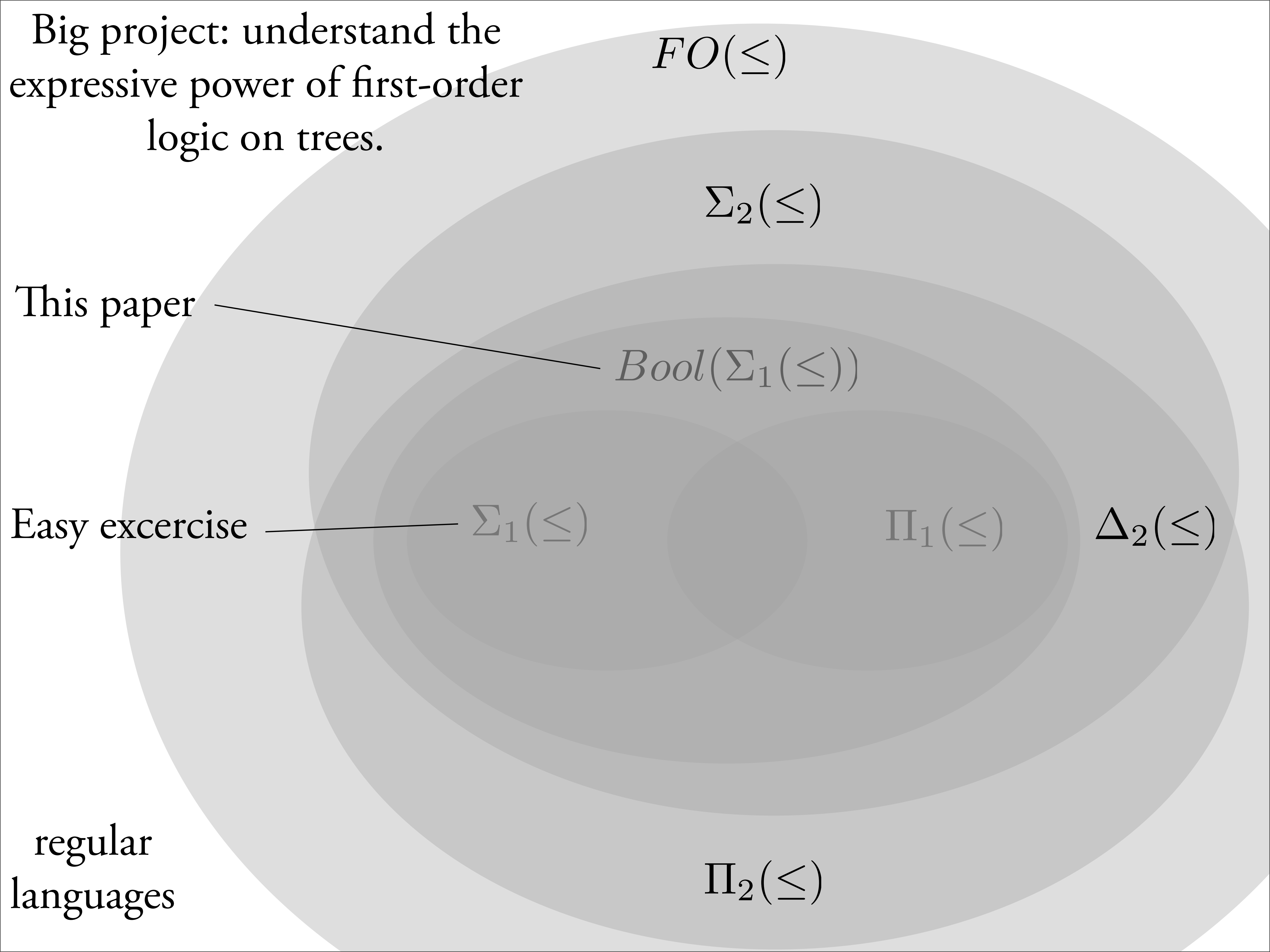
Big project: understand the expressive power of first-order logic on trees.

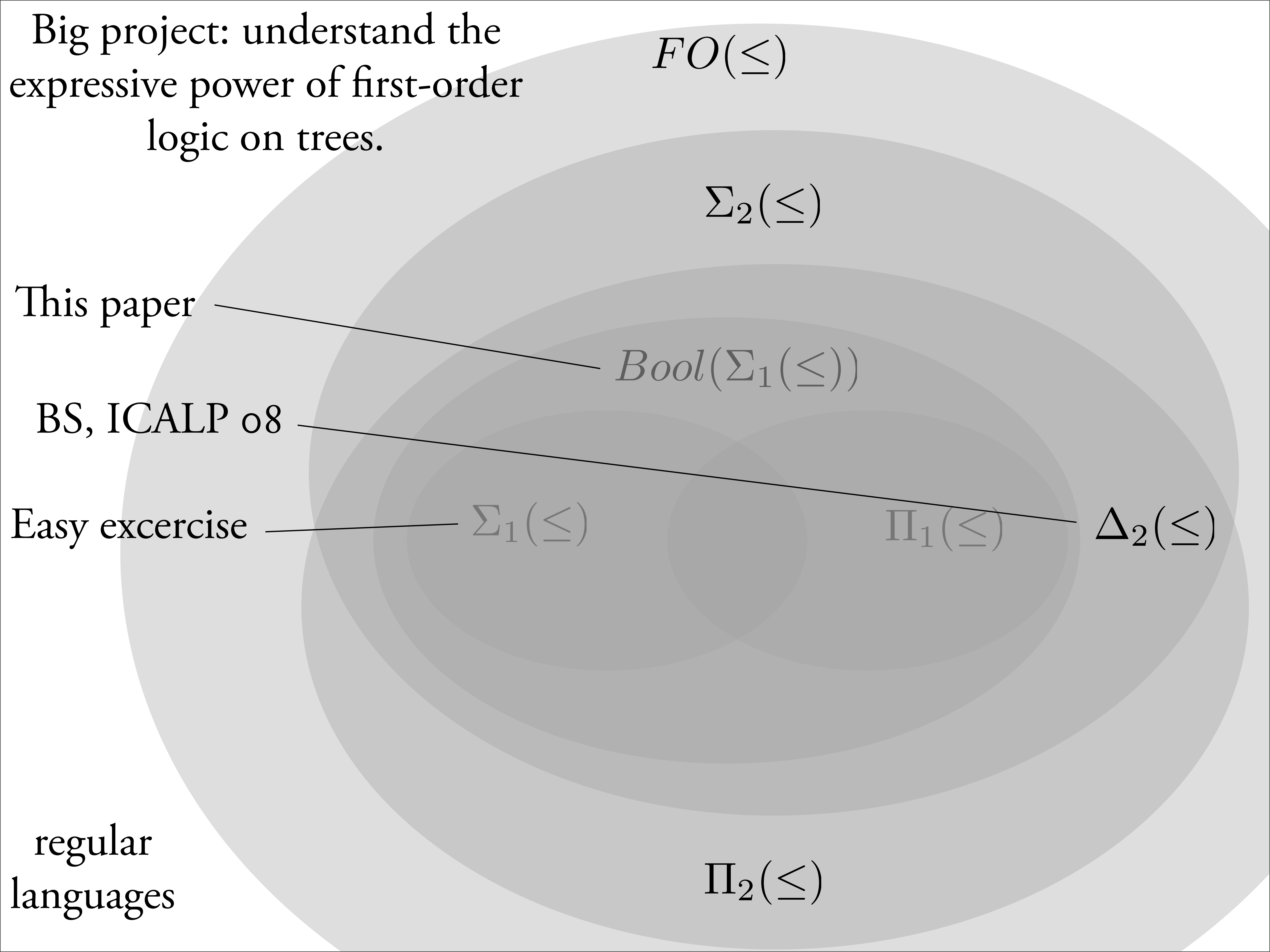


Big project: understand the expressive power of first-order logic on trees.




Big project: understand the expressive power of first-order logic on trees.
$FO(\leq)$
This paper
$Bool(\Sigma_1(\leq))$
Easy excercise
$\Sigma_1(\leq)$
$\Pi_1(\leq)$
regular languages


Big project: understand the expressive power of first-order logic on trees.
$FO(\leq)$
$\Sigma_2(\leq)$
This paper
$Bool(\Sigma_1(\leq))$
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.
$FO(\leq)$
$\Sigma_2(\leq)$
This paper
$Bool(\Sigma_1(\leq))$
BS, ICALP 08
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.