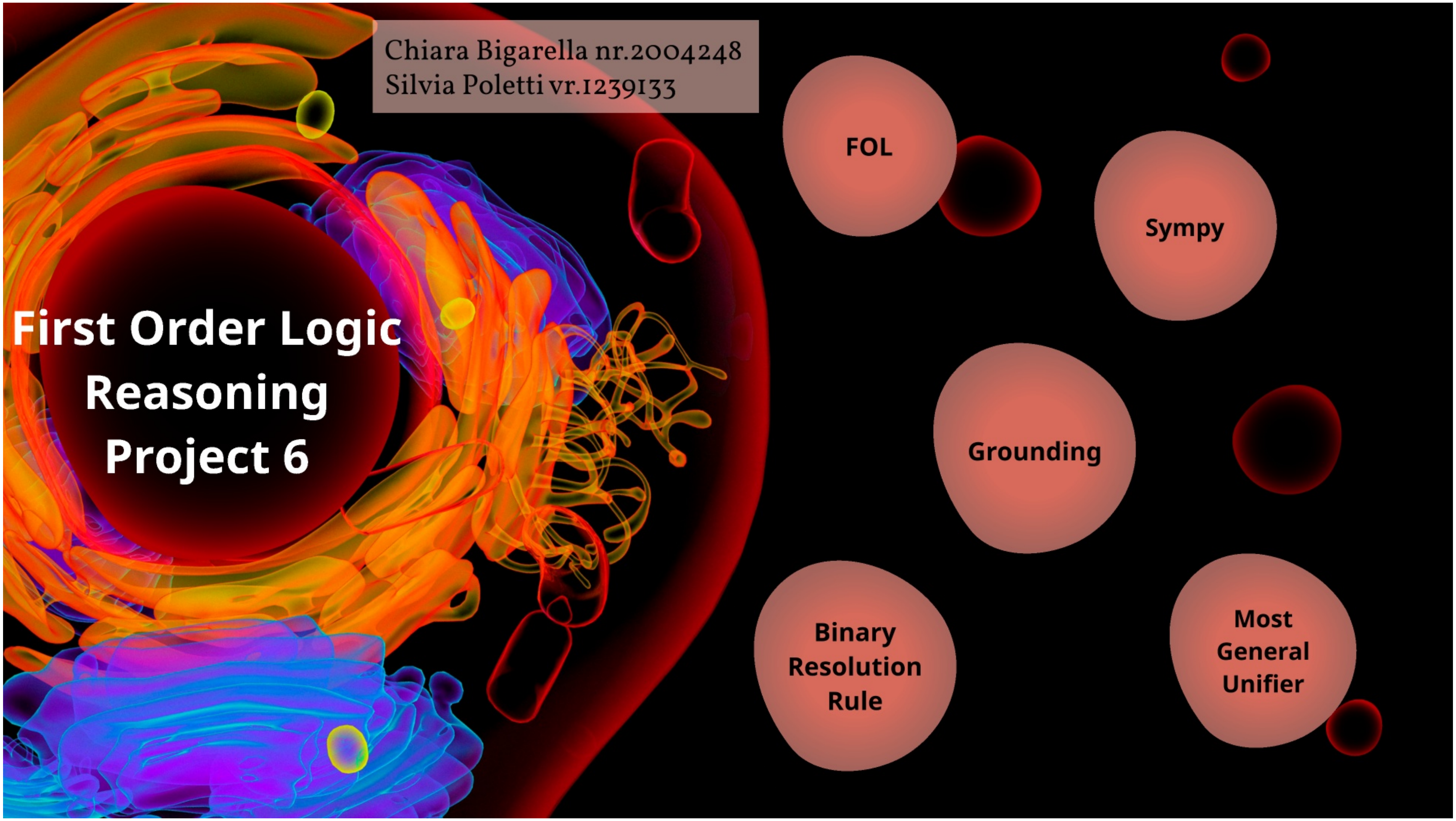
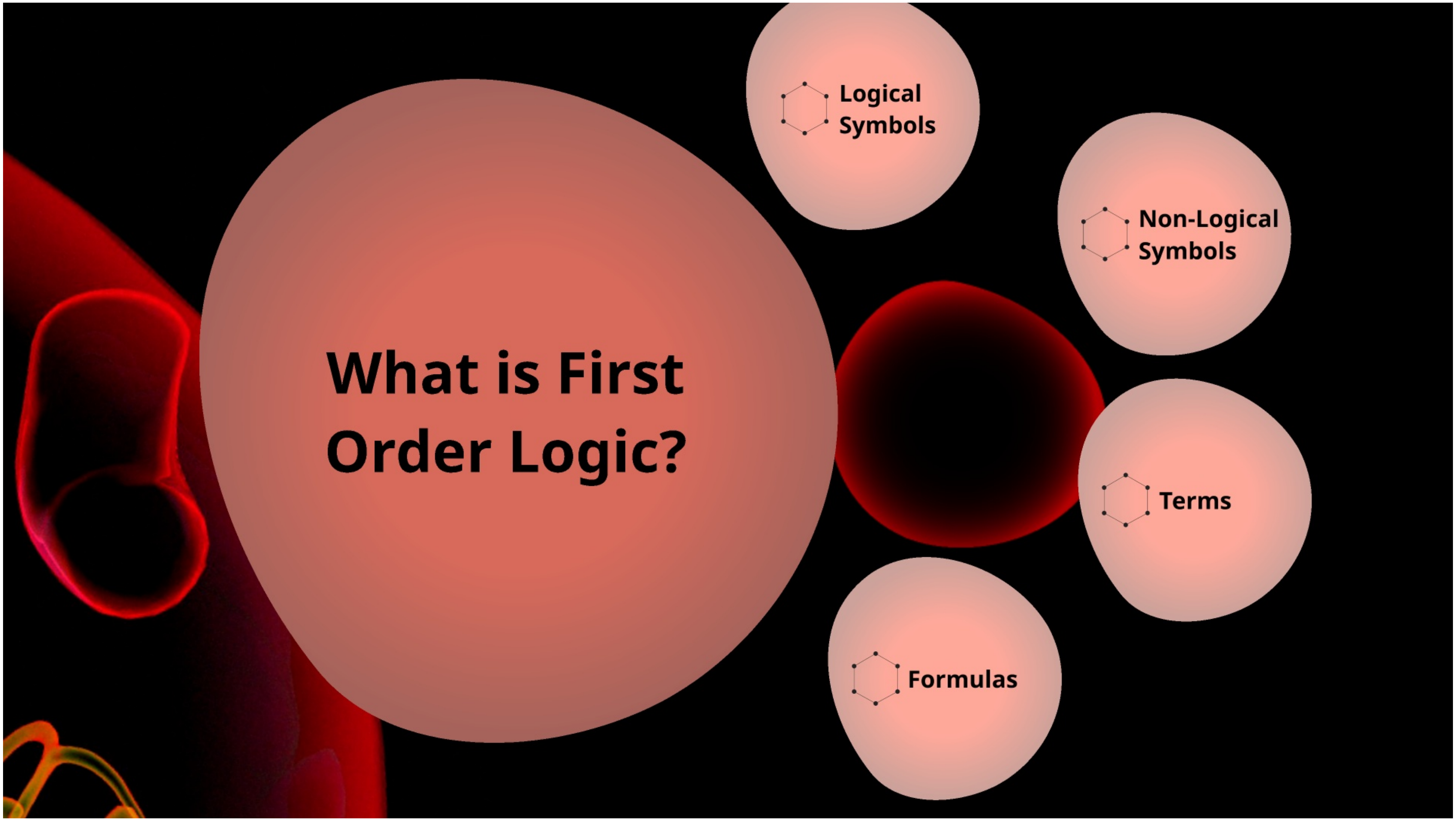

Chiara Bigarella nr.2004248
Silvia Poletti vr.1239133
First Order Logic
Reasoning
Project 6
FOL
Sympy
Grounding
Binary
Resolution
Rule
Most
General
Unifier


What is First Order Logic?
Logical Symbols
Non-Logical Symbols
Terms
Formulas

Logical Symbols

- Logical constants $\bot, \top$
- Propositional logical connectives $\wedge, \vee, \neg, \rightarrow, \equiv$
- Quantifiers $\forall, \exists$
- Variable symbols $x_1, x_2, \ldots$

Non-Logical Symbols

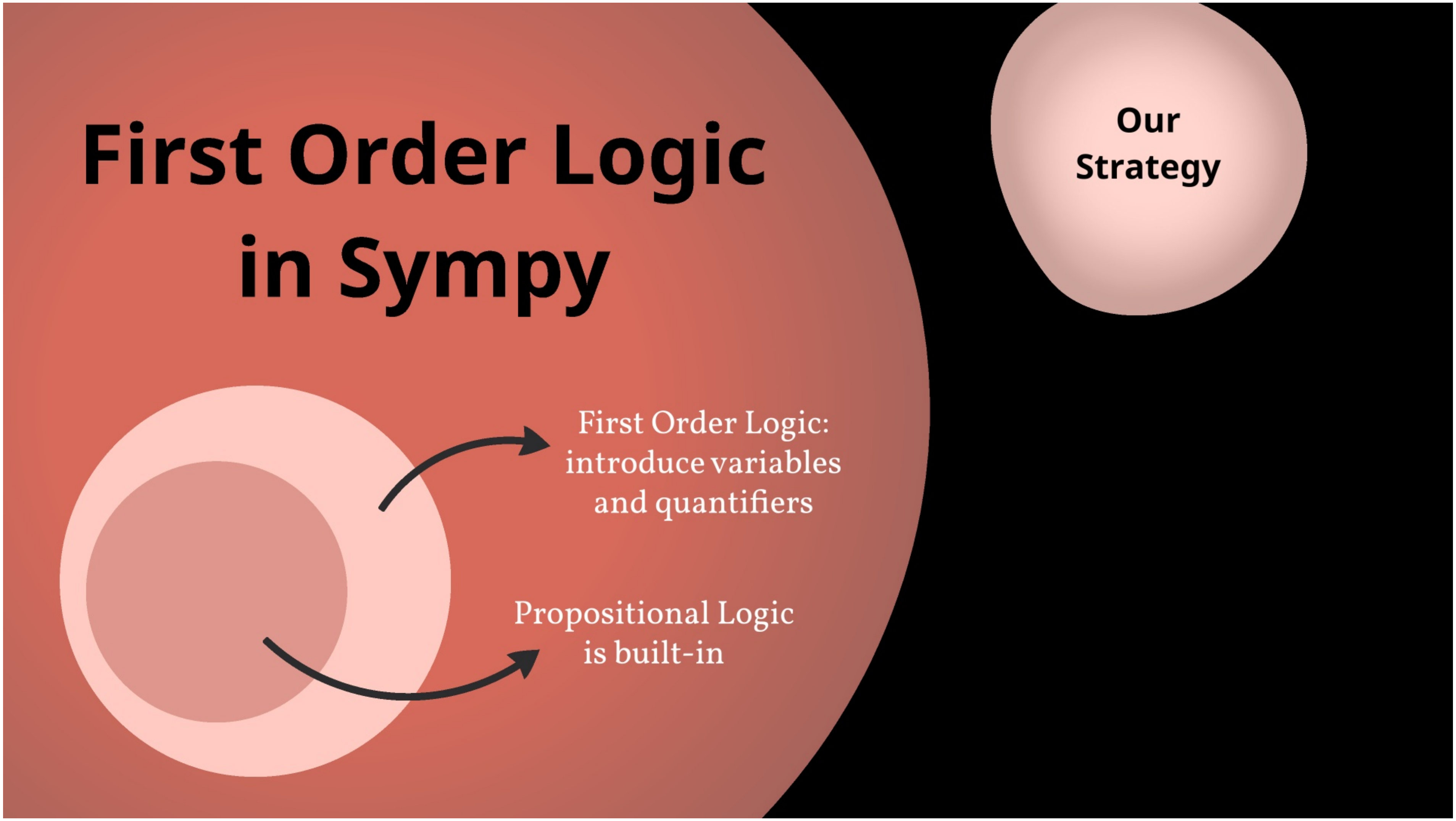
- Constants symbols $c_1, c_2, \ldots$
- Functional symbols $f_1, f_2, \ldots$
- Relational symbols $P_1, P_2, \ldots$
  also called Predicates

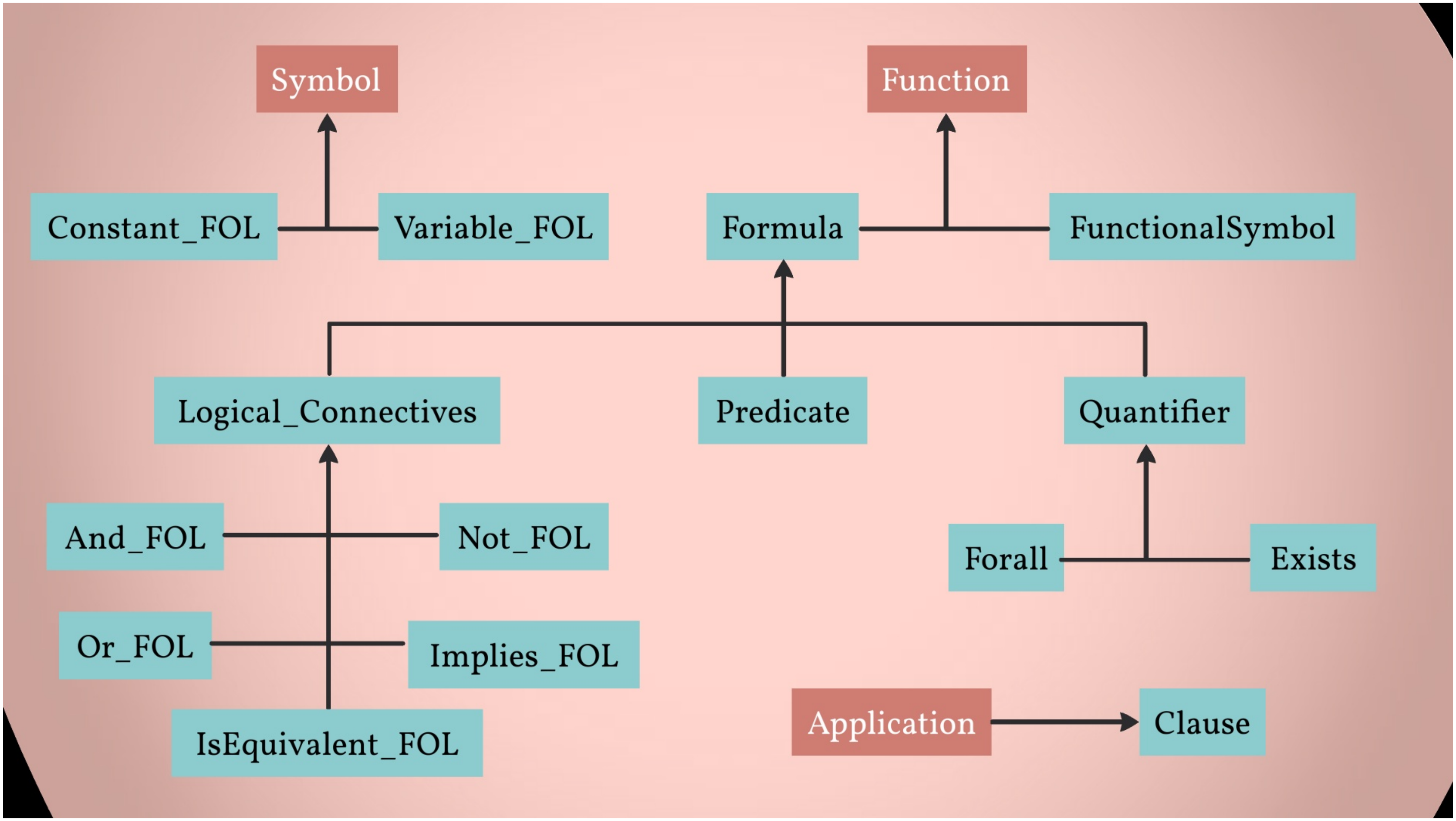
Terms

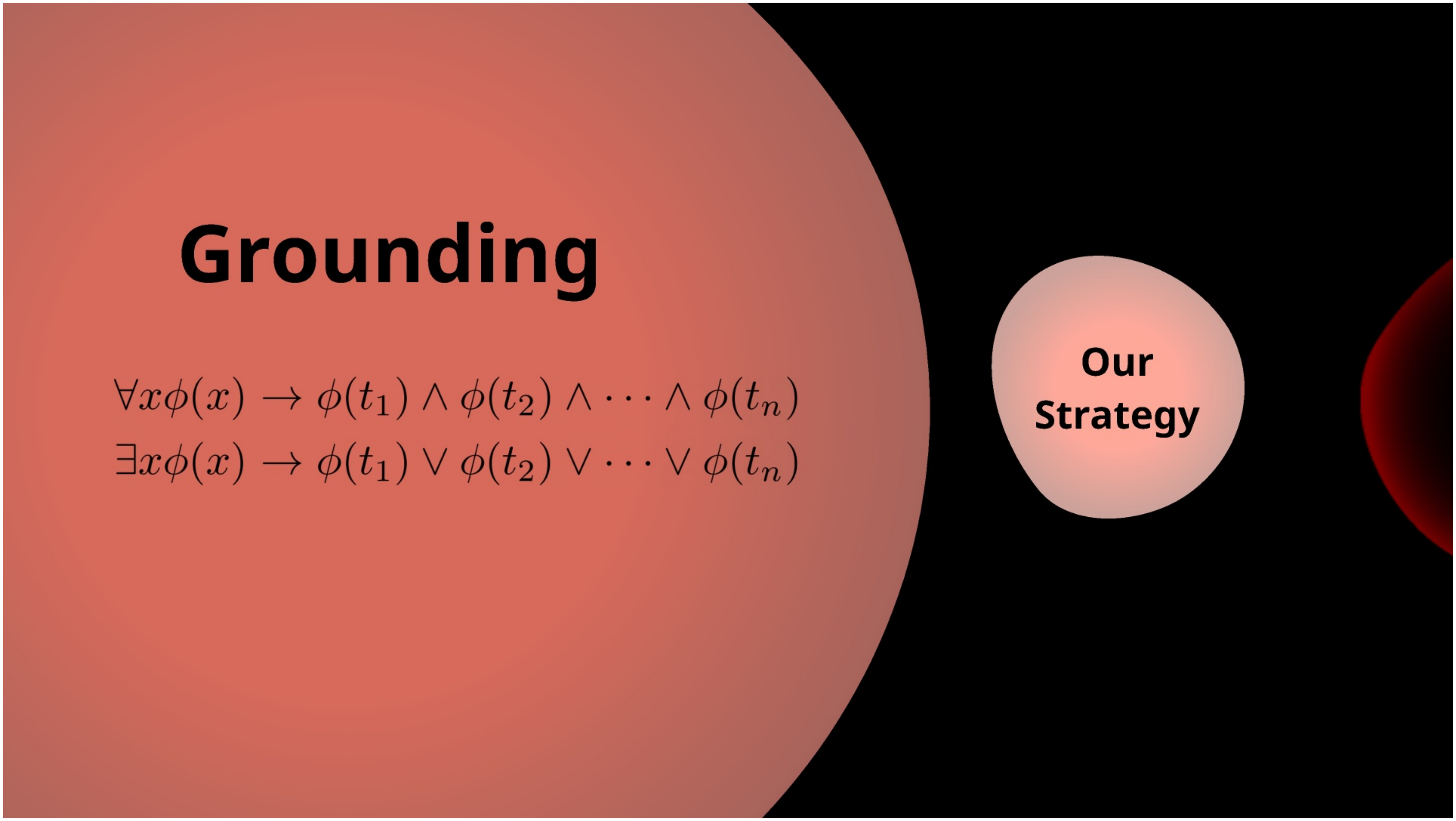
- Constants symbols
  Mary, Biology
- Variable symbols
  x, y
- Functional symbols on constants or variables
  Mother-Of(Mary)
  Mother-Of(Friend-Of(x))
  Mark(Biology, Mary)

Formulas

- Equality between terms
  $Mary = Mother\text{-}Of(John)$
- Predicates applied to the terms
  $Student(x)$
  $Attend(Mary, Biology)$
- Propositional logical connectives on formulas
  $Student(x) \wedge Attend(x, Biology)$
- Quantifiers on formulas
  $\exists x\, (\, Student(x) \wedge Attend(x, Biology)\,)$


First Order Logic
in Sympy
First Order Logic:
introduce variables
and quantifiers
Propositional Logic
is built-in
Our
Strategy


Symbol
Function
Constant_FOL
Variable_FOL
Formula
FunctionalSymbol
Logical_Connectives
Predicate
Quantifier
And_FOL
Not_FOL
Forall
Exists
Or_FOL
Implies_FOL
Application
Clause
IsEquivalent_FOL


Grounding
$\forall x \phi(x) \rightarrow \phi(t_1) \wedge \phi(t_2) \wedge \cdots \wedge \phi(t_n)$
$\exists x \phi(x) \rightarrow \phi(t_1) \vee \phi(t_2) \vee \cdots \vee \phi(t_n)$
Our Strategy

Grounding Implementation

Assumptions:

- Different symbols for each quantifier
- Negated Normal Form
- Prenex Normal Form

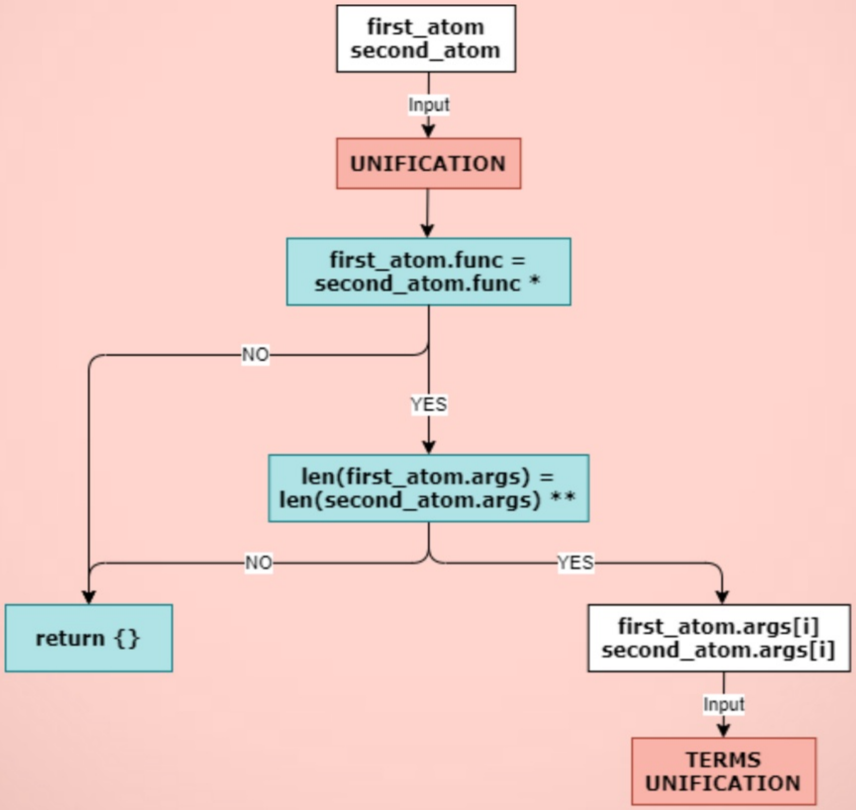
Classes which inherit from Formula
Methods: • _latex • _apply_not • to_nnf • ground

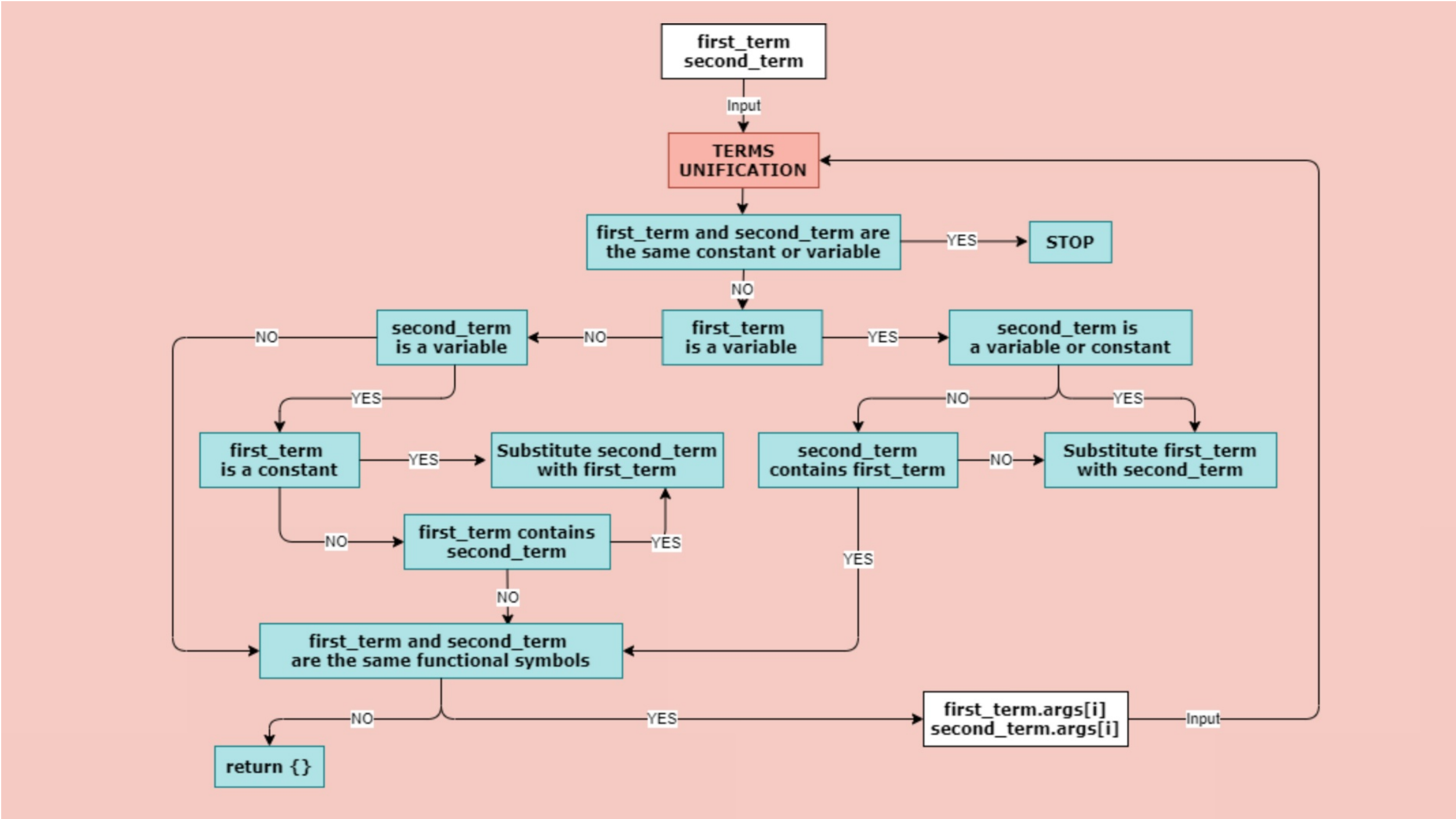
Most General Unifier

- σ is a unifier of terms t and u if $t\sigma = u\sigma$
- σ is more general than θ if $\theta = \sigma \circ \phi$ for some substitution ϕ
- σ is a most general unifier of terms t and u if it is a unifier and is more general than all the other unifiers

Our Strategy

* The attribute *.func* returns the name of the predicate/functional symbol.
** The attribute .args returns the list of arguments of the formula.



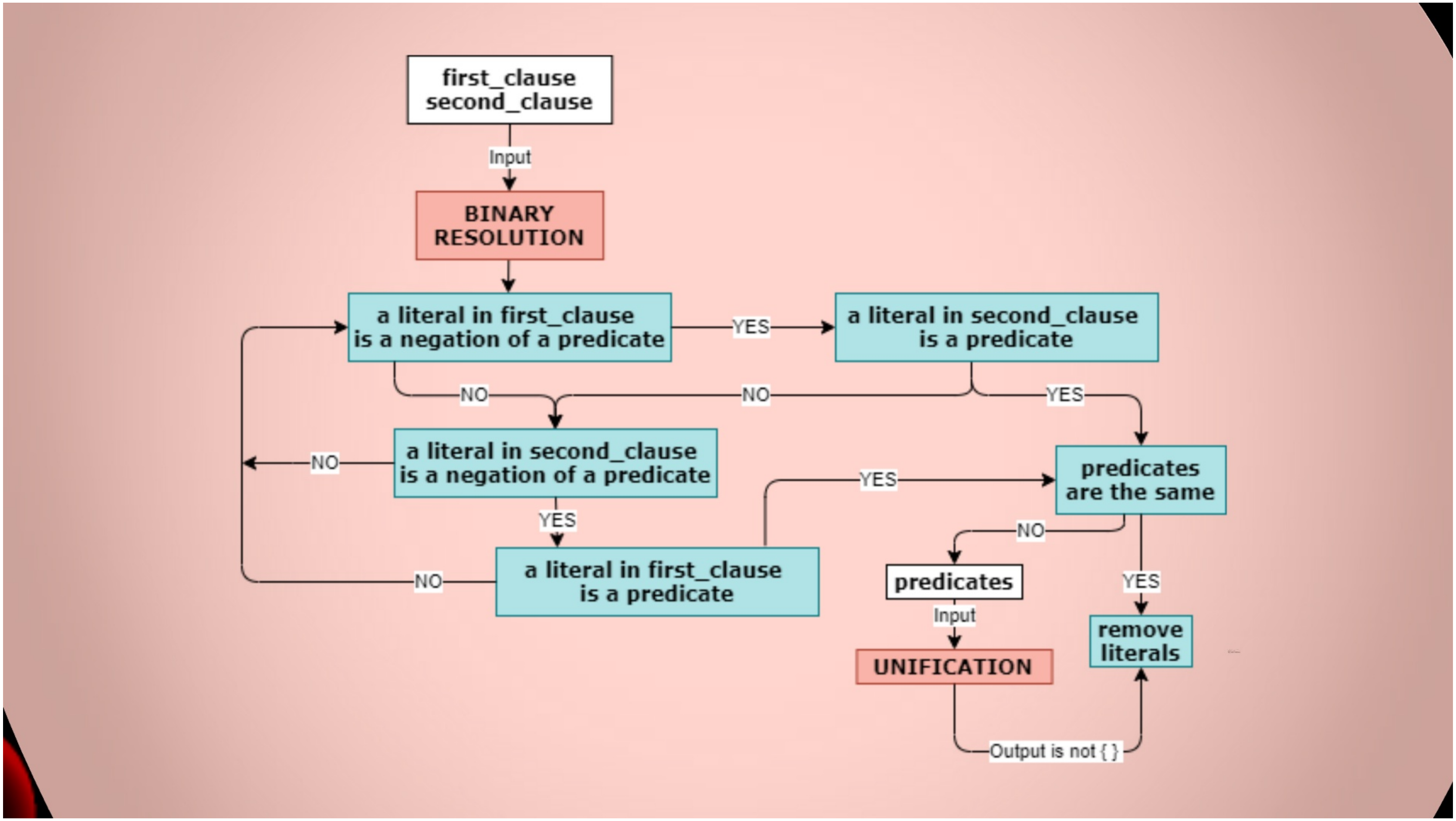

first_term
second_term
Input
TERMS UNIFICATION
first_term and second_term are the same constant or variable
YES
STOP
NO
first_term is a variable
NO
second_term is a variable
YES
second_term is a variable or constant
NO
YES
first_term is a constant
YES
Substitute second_term with first_term
second_term contains first_term
NO
Substitute first_term with second_term
NO
first_term contains second_term
YES
YES
NO
first_term and second_term are the same functional symbols
NO
YES
return {}
first_term.args[i]
second_term.args[i]
Input

Binary Resolution Rule

$$\frac{\{l_1, \ldots, l_n, P(t_1, \ldots, t_n)\}\{\neg P(u_1, \ldots, u_n), l_{n+1}, \ldots, l_m\}}{\{l_1, \ldots, l_m\}\sigma}$$

where l_i is a literal and σ is the Most General Unifier of $P(t_1, \ldots, t_n)$ and $P(u_1, \ldots, u_n)$.

Our Strategy


first_clause
second_clause
Input
BINARY RESOLUTION
a literal in first_clause is a negation of a predicate
YES
a literal in second_clause is a predicate
NO
NO
YES
a literal in second_clause is a negation of a predicate
NO
YES
predicates are the same
YES
a literal in first_clause is a predicate
NO
NO
predicates
Input
UNIFICATION
YES
remove literals
Output is not { }

After considering all the possible couples of literals:

- If we did not perform any unification or any removals then Binary Resolution does not apply
- Otherwise, if we did not end up with empty clauses, we recursively apply the Binary Resolution algorithm on the resulting clauses. At the end we merge the clauses.