Chapter 2.
Declarative Semantics

- Last updated: May 29, 2017 -

How do we know what a goal / program means?

→ Translation of Prolog to logical formulas

How do we know what a logical formula means?

→ Models of logical formulas (Declarative semantics) ← Now

→ Proofs of logical formulas (Operational semantics) ← Later

Question

Question

- What is the meaning of this program?

```
bigger(elephant, horse).
bigger(horse, donkey).
is_bigger(X, Y) :- bigger(X, Y).
is_bigger(X, Y) :- bigger(X, Z), is_bigger(Z, Y).
```

Rephrased question: Two steps

1. How does this program translate to logic formulas?
2. What is the meaning of the logic formulas?

Semantics: Translation

How do we translate a Prolog program to a formula in First Order Logic (FOL)?

→ Translation Scheme

Can any FOL formula be expressed as a Prolog Program?

→ Normalization Steps

Translation of Prolog Programs

1. A Prolog program is translated to a set of formulas, with each clause in the program corresponding to one formula:

   $\{$ bigger(elephant, horse),
   bigger(horse, donkey),
   $\forall x.\forall y.(\text{ bigger}(x, y) \rightarrow \text{is\_bigger}(x, y))$,
   $\forall x.\forall y.(\exists z.(\text{bigger}(x, z) \wedge \text{is\_bigger}(z, y)) \rightarrow \text{is\_bigger}(x, y))$
   $\}$

2. Such a set is to be interpreted as the conjunction of all the formulas in the set:

   bigger(elephant, horse) $\wedge$
   bigger(horse, donkey) $\wedge$
   $\forall x.\forall y.(\text{ bigger}(x, y) \rightarrow \text{is\_bigger}(x, y)) \wedge$
   $\forall x.\forall y.(\exists z.(\text{bigger}(x, z) \wedge \text{is\_bigger}(z, y)) \rightarrow \text{is\_bigger}(x, y))$

Translation of Clauses

- Each comma separating subgoals becomes $\wedge$ (conjunction).
- Each :- becomes $\leftarrow$ (implication)
- Each variable in the head of a clause is bound by a $\forall$ (universal quantifier)
  - `son(X,Y) :- father(Y,X),male(X).`
  - $\forall x.\forall y\ \mathrm{son}(x,y) \leftarrow \mathrm{father}(y,x) \wedge \mathrm{male}(x)$
- Each variable that occurs only in the body of a clause is bound by a $\exists$ (existential quantifier)
  - `grandfather(X):-      father(X,Y) , parent(Y,Z).`
  - $\forall x.\ (\mathrm{grandfather}(x) \leftarrow \exists y.\exists z.\ \mathrm{father}(x,y) \wedge \mathrm{parent}(y,z))$

Translating Disjunction

- Disjunction is the same as two clauses:

```
disjunction(X) :-
    ( ( a(X,Y), b(Y,Z) )
    ; ( c(X,Y), d(Y,Z) )
    ).
```

⇔

```
disjunction(X) :-
    a(X,Y), b(Y,Z).
disjunction(X) :-
    c(X,Y), d(Y,Z) .
```

- Variables with the same name in different clauses are different
- Therefore, variables with the same name in different disjunctive branches are different too!
- Good Style: Avoid accidentally equal names in disjoint branches!
  - Rename variables in each branch and use explicit unification

```
disjunction(X) :-
    ( (X=X1, a(X1,Y1), b(Y1,Z1) )
    ; (X=X2, c(X2,Y2), d(Y2,Z2) )
    ).
```

⇔

```
disjunction(X1) :-
    a(X1,Y1), b(Y1,Z1).
disjunction(X2) :-
    c(X2,Y2), d(Y2,Z2) .
```

Declarative Semantics – in a nutshell

Meaning of Programs (in a nutshell)

Meaning of a program

Meaning of the equivalent formula.

$bigger(elephant, horse)$
$\wedge$
$bigger(horse, donkey)$
$\wedge$
$\forall x.\forall y.(bigger(x, y) \rightarrow is\_bigger(x, y))$
$\wedge$
$\forall x.\forall y.(\exists z.(bigger(x, z) \wedge is\_bigger(z, y)) \rightarrow is\_bigger(x, y))$

Meaning of a formula

Set of logical consequences

$bigger(elephant, horse)$
$\wedge$
$bigger(horse, donkey)$
$\wedge$
$is\_bigger(elephant, horse)$
$\wedge$
$is\_bigger(horse, donkey)$
$\wedge$
$is\_bigger(elephant, donkey)$

Meaning of Programs

Model =
Set of logical consequences =
What is true according to the formula

Meaning of a program

Meaning of the equivalent formula.

bigger(elephant, horse)

$\wedge$

bigger(horse, donkey)

$\wedge$

$\forall x.\forall y.($ bigger(x, y) $\rightarrow$ is_bigger(x, y))

$\wedge$

$\forall x.\forall y.(\exists z.($bigger(x, z) $\wedge$ is_bigger(z, y)) $\rightarrow$ is_bigger(x, y))

Meaning of a formula

Set of logical consequences

bigger(elephant, horse)

$\wedge$

bigger(horse, donkey)

$\wedge$

is_bigger(elephant, horse)

$\wedge$

is_bigger(horse, donkey)

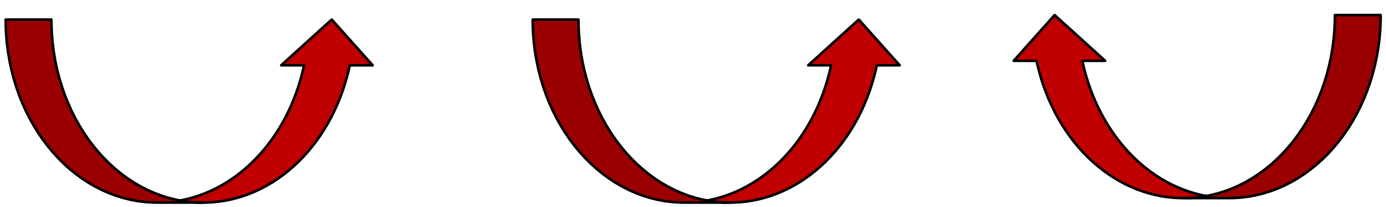
$\wedge$

is_bigger(elephant, donkey)

Semantics of Programs and Queries (in a nutshell)

Program	**Formula**	**Model**	**Query**

`bigger(elephant,horse).`
`bigger(horse,donkey).`
`is_bigger(X,Y) :-`
`   bigger(X,Y).`
`is_bigger(X,Y) :-`
`   bigger(X,Z),`
`   is_bigger(Z,Y).`
	bigger(elephant, horse)
	$\wedge$
	bigger(horse, donkey)
	$\wedge$
	$\forall x.\forall y.(\text{is\_bigger}(x, y) \leftarrow \text{bigger}(x, y))$
	$\wedge$
	$\forall x.\forall y.(\exists z.(\text{is\_bigger}(x, y) \leftarrow \text{bigger}(x, z) \wedge \text{is\_bigger}(z, y)))$
		bigger(elephant, horse)
		$\wedge$
		bigger(horse, donkey)
		$\wedge$
		is_bigger(elephant, horse)
		$\wedge$
		is_bigger(horse, donkey)
		$\wedge$
		is_bigger(elephant, donkey)
			?-
			bigger(elephant, X)
			$\wedge$
			is_bigger(X, donkey)



Translation **Interpretation** (logical consequence) **Matching**

Model-based Semantics → Algorithm

Model-based semantics

- Herbrand interpretations and Herbrand models
- Basic step = “Entailment” (Logical consequence)
- A formula is true if it is a logical consequence of the program

Algorithm = Logic + Control

- Logic = Clauses
- Control =
  - Bottom-up fixpoint iteration to build the model
  - Matching of queries to the model

Program

```
bigger(elephant,horse).
bigger(horse,donkey).
...
```

Formula

bigger(elephant, horse)
^
bigger(horse, donkey)
^
…

Model

bigger(elephant, horse)
^
bigger(horse, donkey)
^
…

Query

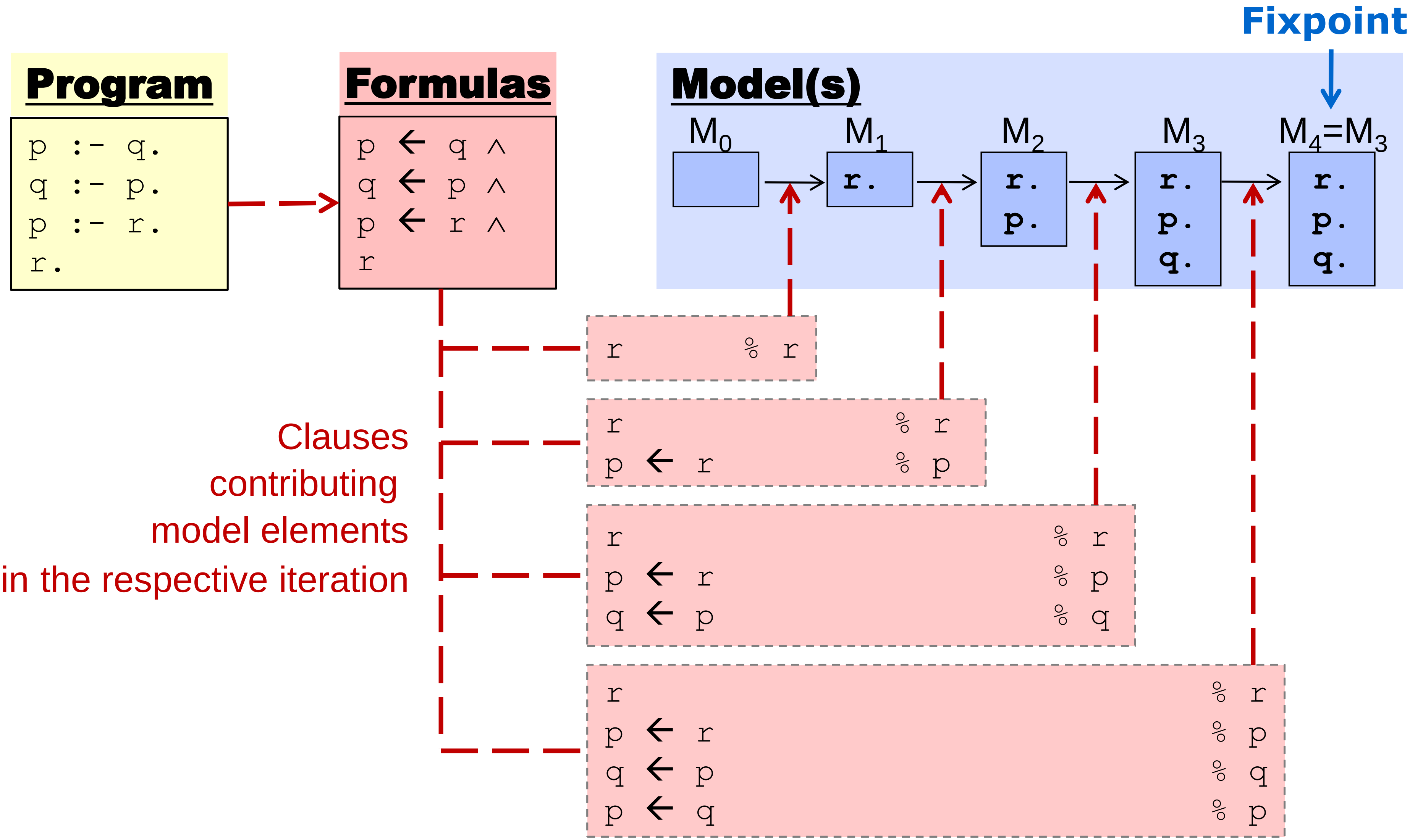
?-
bigger(elephant, X)
^
is_bigger(X, donkey)

Translation (Program → Formula)

Interpretation (logical consequence) (Formula → Model)

Matching (Query → Model)

Constructing Models by Fixpoint Iteration



universitätbonn

Declarative Semantics Assessed

Pro

- Simple
  - Easy to understand
- Thorough formal foundation
  - implication (entailment)

Perfect for understanding the meaning of a program

Excellent query language

Contra

- Inefficient
  - Need to build the whole model in the worst case
- Inapplicable to infinite models
  - Never terminates if the query is not true in the model

Bad as the basis of a practical interpreter implementation

Cannot express execution order, side-effects (e.g. I/O), …

No programming language

Chapter Summary

- Translation to logic
  - From clauses to formulas
- Declarative / Model-based Semantics
  - Herbrand Universe
  - Herbrand Interpretation
  - Herbrand Model
- Operational interpretation
  - Model construction by fix-point iteration
  - Matching of goals to the model
- Assesssment
  - Strength
  - Weaknesses