***Auxiliary material - related to predicate calculus and undecidability.***

# Clausal logic

Peter Flach

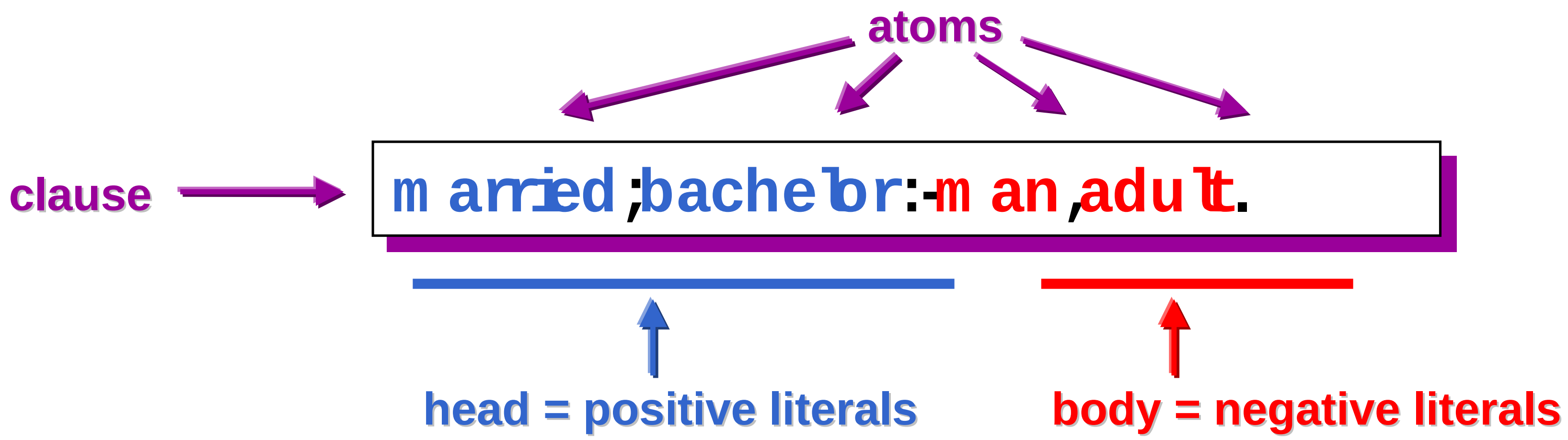
University of Bristol

- ☞ ***Propositional* clausal logic**
  - ✓expressions that can be true or false
- ☞ ***Relational* clausal logic**
  - ✓constants and variables refer to objects
- ☞ ***Full* clausal logic**
  - ✓functors aggregate objects
- ☞ ***Definite clause* logic = pure Prolog**
  - ✓no disjunctive heads

## Clausal logic

# Propositional clausal logic: syntax

"Somebody is married **or** a bachelor **if** he is a man **and** an adult."



```
married;bachelor:-man,adult.
```

$$\boxed{\texttt{married} \lor \texttt{bachelor} \lor \neg\texttt{man} \lor \neg\texttt{adult}}$$

- Persons are happy or sad

```
happy;sad:-person.
```

- No person is both happy and sad

```
:-person,happy,sad.
```

- Sad persons are not happy

```
:-person,sad,happy.
```

- Non-happy persons are sad

```
sad;happy:-person.
```

## Exercise 2.1

- ***Herbrand base***: set of atoms

  `{married,bachelor,man,adult}`

- ***Herbrand interpretation***: set of **true** atoms

  `{married,man,adult}`

- A clause is **false** in an interpretation if all body-literals are **true** and all head-literals are **false**…

  `bachelor:-man,adult.`

- …and **true** otherwise: the interpretation is a ***model*** of the clause.

  `:-married,bachelor.`

## Propositional clausal logic: semantics

☞ A clause ***C*** is a ***logical consequence*** of a program (set of clauses) ***P*** iff every model of ***P*** is a model of ***C***.

☞ Let ***P*** be

```
married;bachelor:-man,adult.
man.
:-bachelor.
```

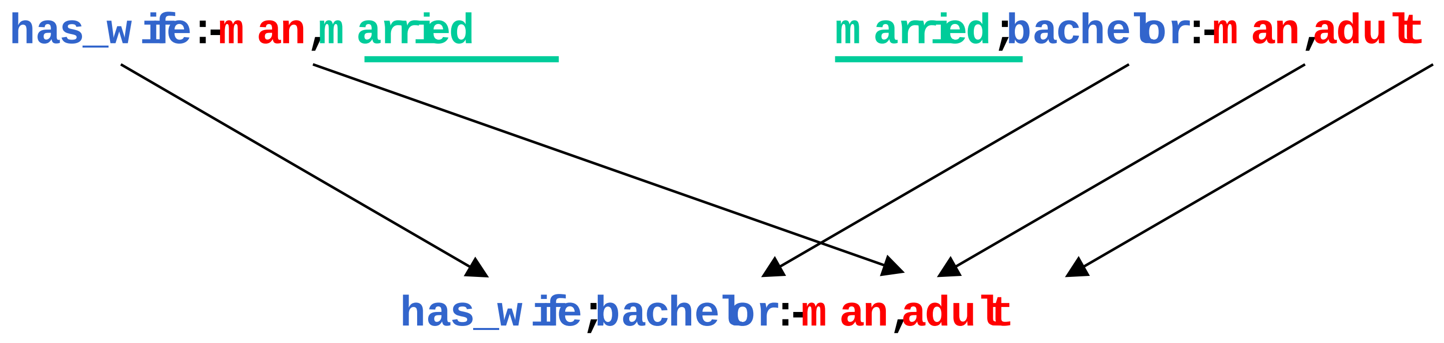
☞ **`married:-adult`** is a logical consequence of ***P***;

☞ **`married:-bachelor`** is a logical consequence of ***P***;

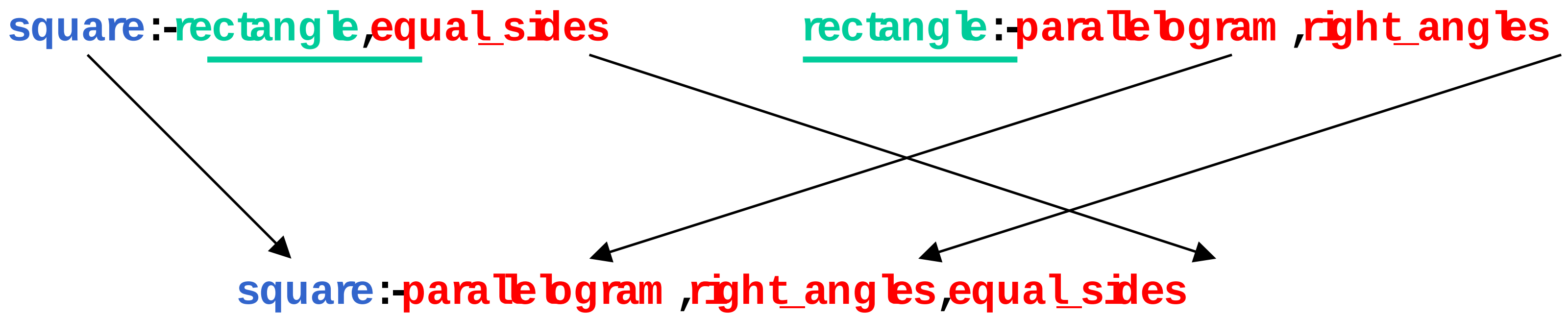
☞ **`bachelor:-man`** is not a logical consequence of ***P***;

☞ **`bachelor:-bachelor`** is a logical consequence of ***P***.

## Exercise 2.2



```
has_wife:-man,married          married;bachelor:-man,adult

            has_wife;bachelor:-man,adult
```



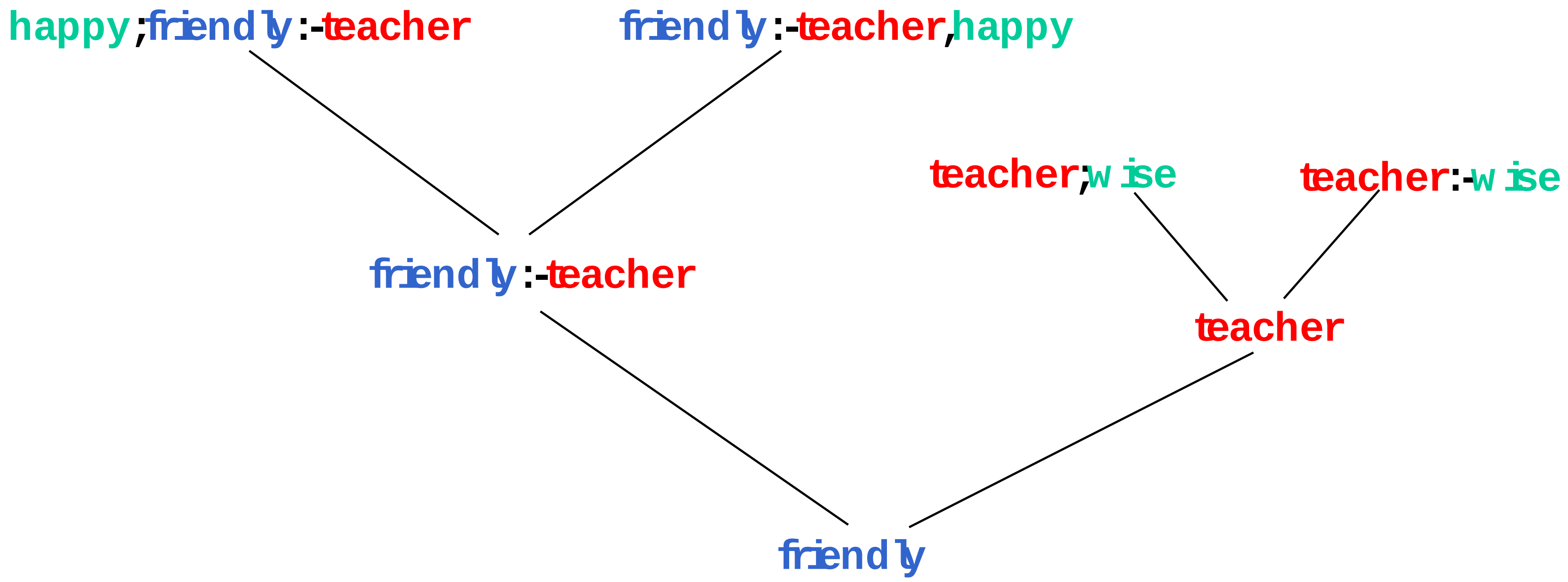
```
square:-rectangle,equal_sides          rectangle:-parallelogram,right_angles

          square:-parallelogram,right_angles,equal_sides
```

# Propositional resolution

- ☞ Propositional resolution is
  - ✓***sound***: it derives only logical consequences.
  - ✓***incomplete***: it cannot derive arbitrary tautologies like a:-a…
  - ✓…but ***refutation-complete***: it derives the empty clause from any inconsistent set of clauses.

- ☞ ***Proof by refutation***: add the negation of the assumed logical consequence to the program, and prove inconsistency by deriving the empty clause.

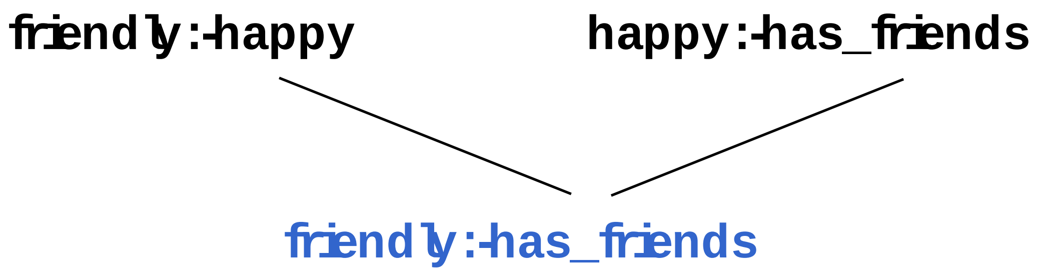
## Propositional clausal logic: meta-theory

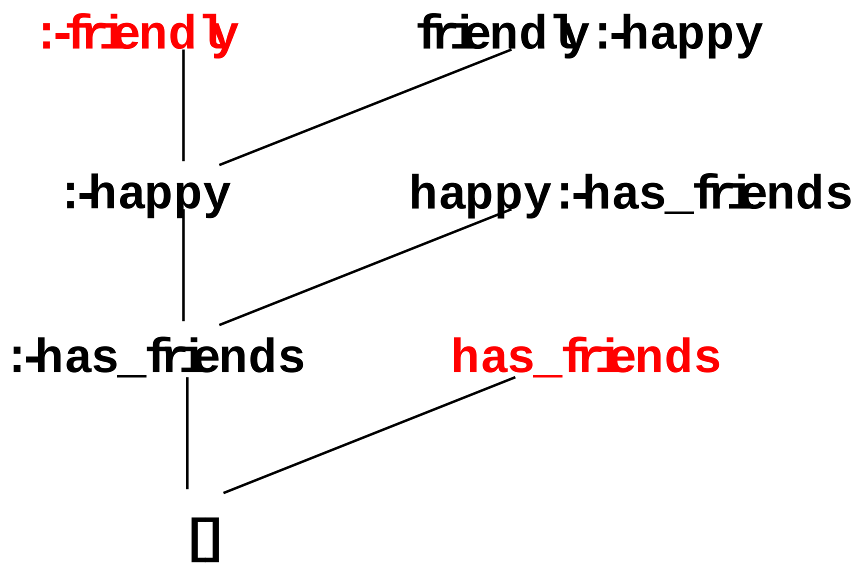

happy;friendly:-teacher
friendly:-teacher,happy
teacher;wise
teacher:-wise
friendly:-teacher
teacher
friendly

## Exercise 2.4

## Exercise 2.5

### Direct proof:

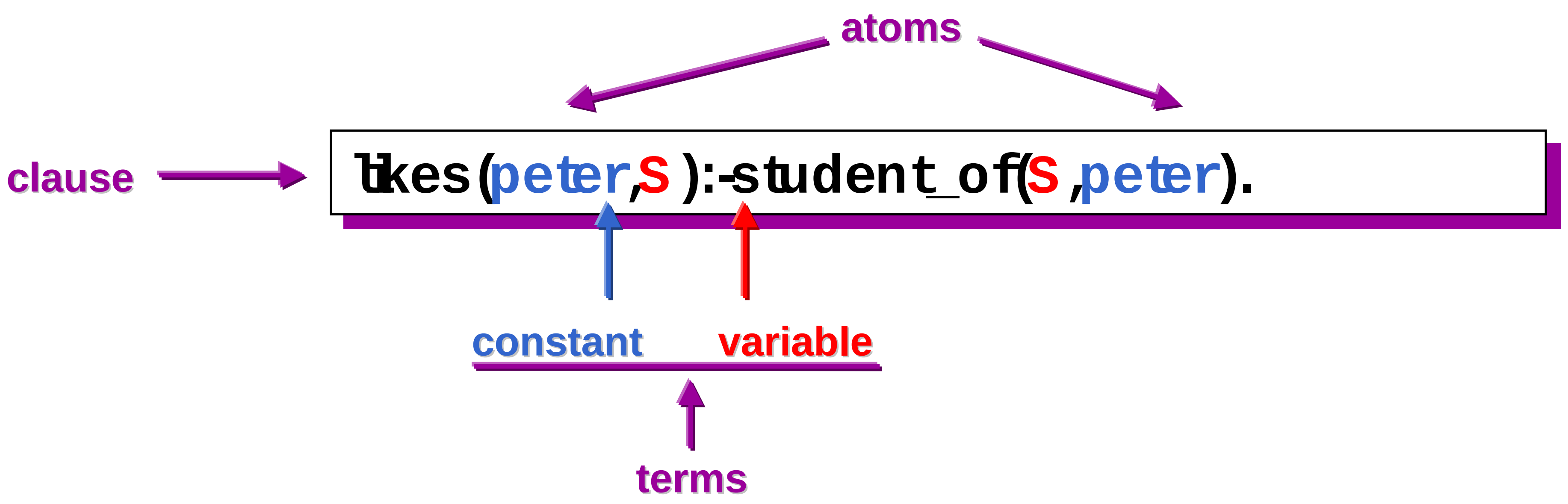




### Proof by refutation:

¬(friendly:-has_friends) ⇒
¬(friendly ∨ ¬has_friends) ⇒
(¬friendly) ∧ (has_friends) ⇒
:-friendly and has_friends

"Peter likes anybody who is his student."



## Relational clausal logic: syntax

- ☞ A ***substitution*** maps variables to terms:

```
{S->maria}
```

- ☞ A substitution can be ***applied*** to a clause:

```
likes(peter,maria):-student_of(maria,peter).
```

- ☞ The resulting clause is said to be an ***instance*** of the original clause, and a ***ground instance*** if it does not contain variables.

- ☞ Each instance of a clause is among its logical consequences.

## Substitutions

- ***Herbrand universe***: set of ground terms (i.e. constants)

  `{peter,maria}`

- ***Herbrand base***: set of ground atoms

  ```
  {likes(peter,peter),likes(peter,maria),likes(maria,peter),
   likes(maria,maria),student_of(peter,peter),student_of(peter,maria),
   student_of(maria,peter),student_of(maria,maria)}
  ```

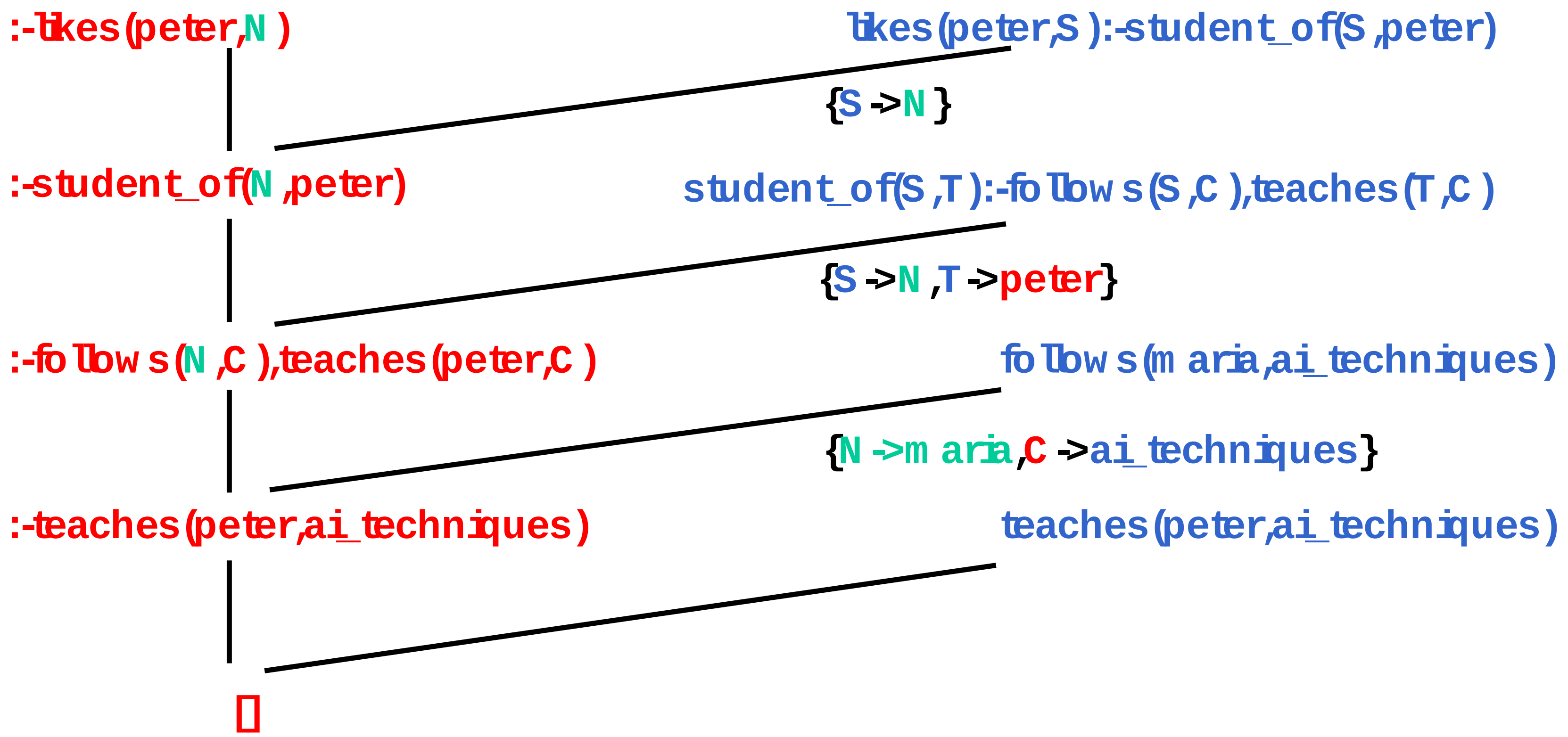
- ***Herbrand interpretation***: set of **true** ground atoms

  `{likes(peter,maria),student_of(maria,peter)}`

- An interpretation is a **model** for a clause if it makes all of its ground instances **true**

  ```
  likes(peter,maria):-student_of(maria,peter).

  likes(peter,peter):-student_of(peter,peter).
  ```

## Relational clausal logic: semantics

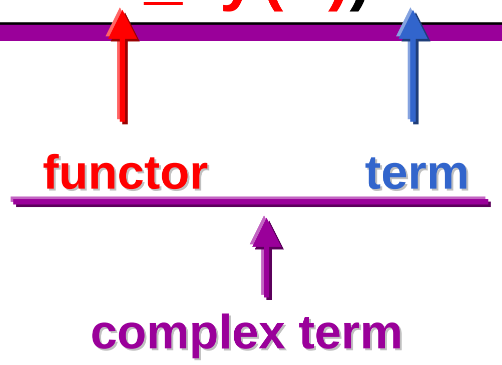


```
:-likes(peter,N)                      likes(peter,S):-student_of(S,peter)
                                      {S->N}
:-student_of(N,peter)                 student_of(S,T):-follows(S,C),teaches(T,C)
                                      {S->N,T->peter}
:-follows(N,C),teaches(peter,C)       follows(maria,ai_techniques)
                                      {N->maria,C->ai_techniques}
:-teaches(peter,ai_techniques)        teaches(peter,ai_techniques)

□
```

# Relational resolution

“Everybody loves somebody.”

```
loves(X,person_loved_by(X)).
```



```
loves(peter,person_loved_by(peter)).
loves(anna,person_loved_by(anna)).
loves(paul,person_loved_by(paul)).
…
```

## Full clausal logic: syntax

☞ Every mouse has a tail

```
tail_of(tail(X),X):-mouse(X).
```

☞ Somebody loves everybody

```
loves(person_who_loves_everybody,X).
```

☞ Every two numbers have a maximum

```
maximum_of(X,Y,max(X,Y)):-number(X),number(Y).
```

## Exercise 2.9

☞ ***Herbrand universe***: set of ground terms

```
{0,s(0),s(s(0)),s(s(s(0))),… }
```

☞ ***Herbrand base***: set of ground atoms

```
{plus(0,0,0), plus(s(0),0,0), … ,
 plus(0,s(0),0), plus(s(0),s(0),0), … ,
 … ,
 plus(s(0),s(s(0)),s(s(s(0)))), … }
```

☞ ***Herbrand interpretation***: set of **true** ground atoms

```
{plus(0,0,0), plus(s(0),0,s(0)), plus(0,s(0),s(0))}
```

☞ Some programs have only infinite models

```
plus(0,X,X).
plus(s(X),Y,s(Z)):-plus(X,Y,Z).
```

## Full clausal logic: semantics

```
plus(X,Y,s(Y))
```
and
```
plus(s(V),W,s(s(V)))
```
unify to
```
plus(s(V),s(V),s(s(V)))
```

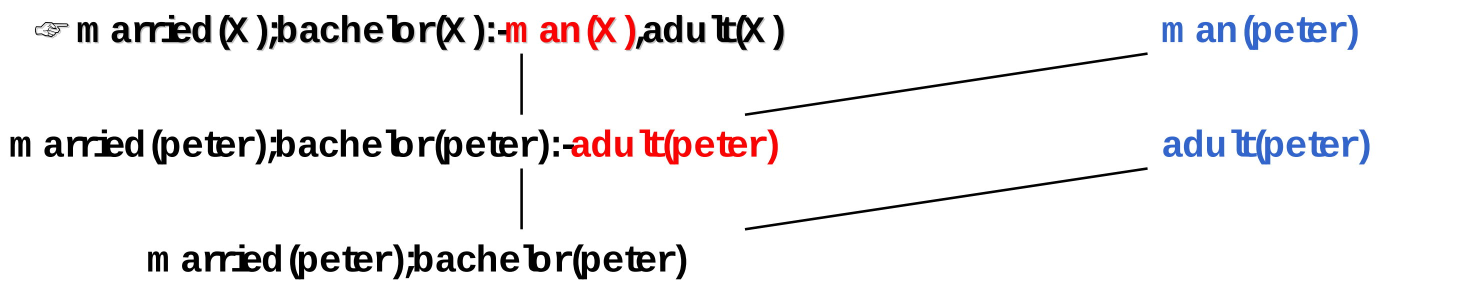
```
length([X|Y],s(0))
```
and
```
length([V],V)
```
unify to
```
length([s(0)],s(0))
```

```
larger(s(s(X)),X)
```
and
```
larger(V,s(V))
```
do **not** unify (occur check!)

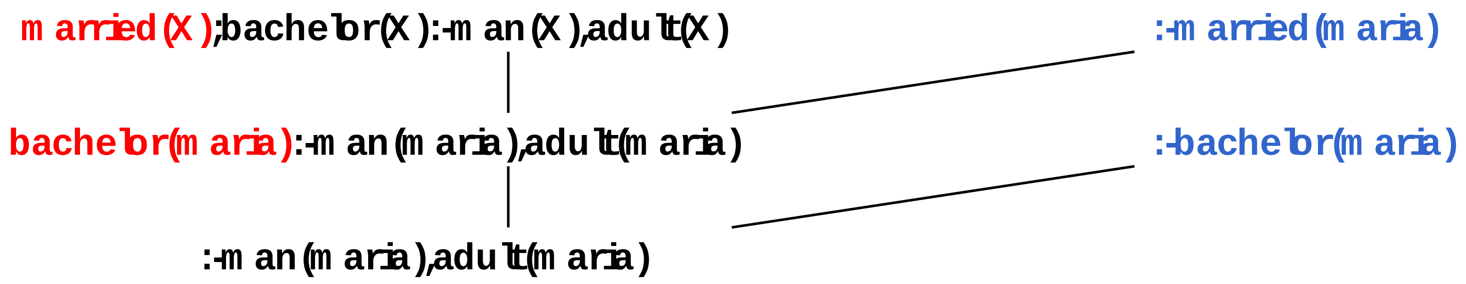
## Exercise 2.11

| | Propositional — | Relational — | Full clausal logic |
|---|---|---|---|
| **Herbrand universe** | — | {a, b}<br>(finite) | {a, f(a), f(f(a)),… }<br>(infinite) |
| **Herbrand base** | {p, q} | {p(a,a), p(b,a),… }<br>(finite) | {p(a,f(a)), p(f(a), f(f(a))),… }<br>(infinite) |
| **clause** | p:-q. | p(X,Z):-q(X,Y),p(Y,Z). | p(X,f(X)):-q(X). |
| **Herbrand models** | ∅<br>{p}<br>{p, q} | ∅<br>{p(a,a)}<br>{p(a,a), p(b,a), q(b,a)}<br>…<br>(finite number of finite models) | ∅<br>{p(a,f(a)), q(a)}<br>{p(f(a), f(f(a))), q(f(a))}<br>…<br>(infinite number of finite or infinite models) |
| **Meta-theory** | sound<br>refutation-complete<br>decidable | sound<br>refutation-complete<br>decidable | sound (if unifying with occur check)<br>refutation-complete<br>semi-decidable |

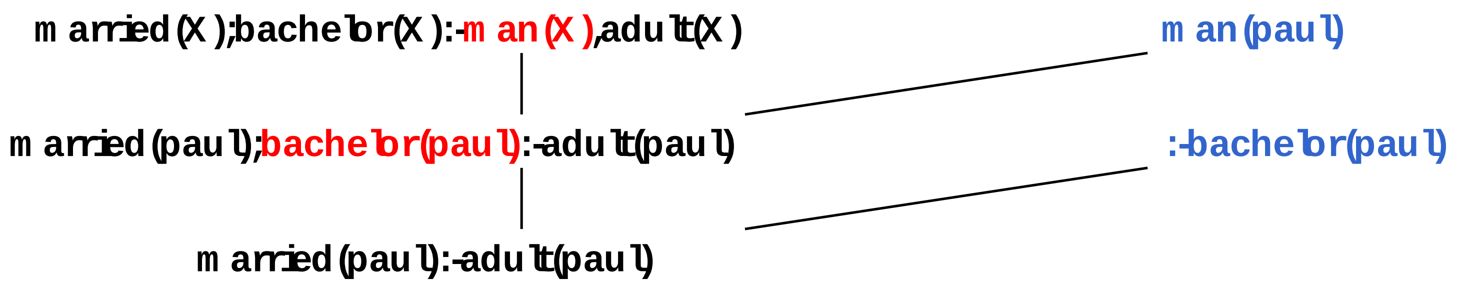
## Summary



```
☞ married(X);bachelor(X):-man(X),adult(X)        man(peter)
married(peter);bachelor(peter):-adult(peter)     adult(peter)
married(peter);bachelor(peter)
```



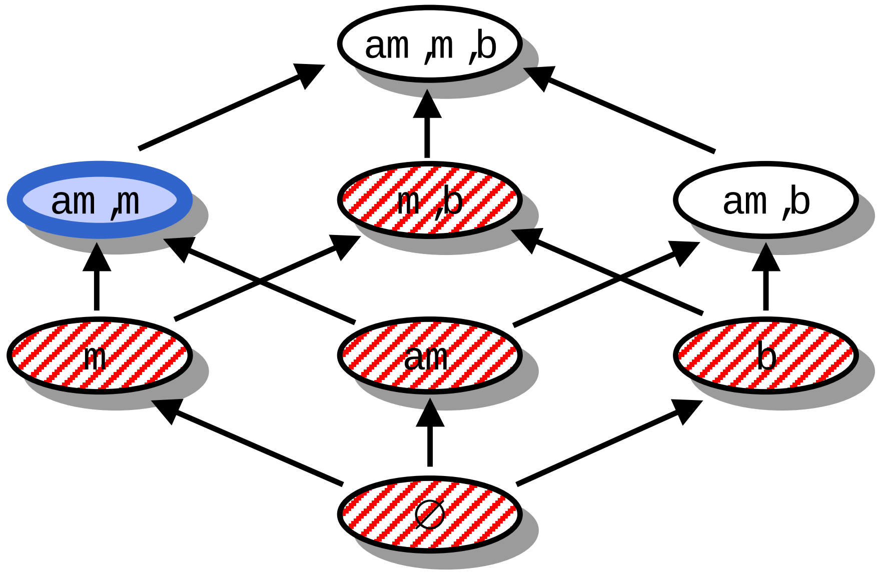
```
married(X);bachelor(X):-man(X),adult(X)          :-married(maria)
bachelor(maria):-man(maria),adult(maria)         :-bachelor(maria)
:-man(maria),adult(maria)
```



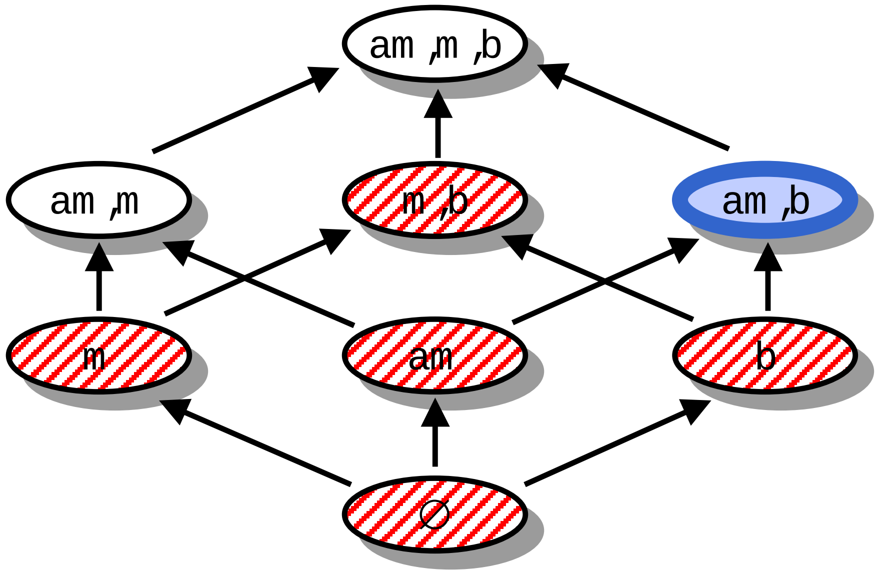
```
married(X);bachelor(X):-man(X),adult(X)          man(paul)
married(paul);bachelor(paul):-adult(paul)        :-bachelor(paul)
married(paul):-adult(paul)
```

# Exercise 2.12

```
married;bachelor:-adult_man.
adult_man.
```



```
married:-adult_man,not bachelor.
```



```
bachelor:-adult_man,not married.
```

## From indefinite to general clauses

- “Everyone has a mother, but not every woman has a child.”

  $\forall Y \exists X\, mother\_of(X,Y) \land \neg \forall Z \exists W\, woman(Z) \rightarrow mother\_of(Z,W)$

- push negation inside

  $\forall Y \exists X\, mother\_of(X,Y) \land \exists Z \forall W\, woman(Z) \land \neg mother\_of(Z,W)$

- drop quantifiers (Skolemisation)

  $mother\_of(mother(Y),Y) \land woman(childless\_woman) \land \neg mother\_of(childless\_woman,W)$

- (convert to CNF and) rewrite as clauses

```
mother_of(mother(Y),Y).
woman(childless_woman).
:-mother_of(childless_woman,W).
```

## From first-order logic to clausal logic