Declarative Programming

2: theoretical backgrounds

Logic Systems: *structure and meta-theoretical properties*

logic system

syntax

defines which "sentences" are legal in the logical language

semantics

gives a meaning to the sentences

usually truth-functional: what is the truth value of a sentence given the truth value of its words

proof theory

specifies how to obtain new sentences (theorems) from assumed ones (axioms) through inference rules

weakest form: prove nothing

soundness

about

anything you can prove is true

completeness

about

anything that is true can be proven

Logic Systems:
roadmap towards Prolog

clausal logic

propositional clausal logic

statements that can be true or false

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

relational clausal logic

statements concern relations among objects from a universe of discourse

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

full clausal logic

compound terms aggregate objects

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

definite clause logic

Pure Prolog

no disjunction in head

lacks control constructs, arithmetic of full Prolog

Propositional Clausal Logic - *Syntax*: clauses

:-	if
;	or
,	and

optional

zero or more

```
clause : head [:- body]
  head : [atom[;atom]*]
  body : atom[,atom]*
  atom : single word starting with lower case
```

"someone is married or a bachelor if he is a man and an adult"

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

Propositional Clausal Logic - *Syntax*: negative and positive literals of a clause

clause

```
H1;...;Hn :- B1,...,Bm
```

$B \Rightarrow H$

$\equiv \neg B \vee H$

is equivalent to

$H1 \vee \ldots \vee Hn \vee \neg B1 \vee \ldots \vee \neg Bm$

positive literals

negative literals

hence a clause can also be defined as a disjunction of literals $L_1 \vee L_2 \vee \ldots \vee L_n$ where each L_i is a literal, i.e. $L_i = A_i$ or $L_i = \neg A_i$, with A_i a proposition.

Propositional Clausal Logic - *Syntax*: logic program

finite set of clauses, each terminated by a period

to be read conjunctively

```
woman;man :- human.
    human :- man.
    human :- woman.
```

is equivalent to

$$
\begin{aligned}
&(human \Rightarrow (woman \vee man)) \\
\wedge &(man \Rightarrow human) \\
\wedge &(woman \Rightarrow human)
\end{aligned}
$$

$$
\begin{aligned}
&(\neg human \vee woman \vee man) \\
\wedge &(\neg man \vee human) \\
\wedge &(\neg woman \vee human)
\end{aligned}
$$

$B \Rightarrow H$

$\equiv \neg B \vee H$

Propositional Clausal Logic - *Syntax*: special clauses

an **empty body** stands for **true**	an **empty head** stands for **false**
`man :-.` or `man.`	`:- impossible.`
`true ⇒ man`	`impossible ⇒ false`

`man ∧ ¬impossible`

Propositional Clausal Logic - *Semantics*: Herbrand base, interpretation and models

Herbrand base B_P of a program P

set of all atoms occurring in P

Herbrand interpretation i of P

when represented by the set of true propositions I: subset of Herband base

mapping from Herbrand base B_P to the set of truth values

```
i : BP → {true, false}
```

An interpretation is a **model for a clause** if the clause is true under the interpretation.

if either the head is true or the body is false

H	B	H:-B
true	true	true
false	true	false
true	false	true
false	false	true

An interpretation is a **model for a program** if it is a model for each clause in the program.

Propositional Clausal Logic - *Semantics*: example (1)

program P

```
woman;man :- human.
human :- man.
human :- woman.
```

Herbrand base B_P

```
{woman,man,human}
```

2^3 possible Herbrand Interpretations

```
I={woman}
```

```
J={woman, man}
```

```
K={woman, man, human}
```

```
L={man}
```

```
M={man, human}
```

```
N={human}
```

```
O={woman, human}
```

```
n={(woman, false),
(man, false),
(human, false)}
```

```
P=∅
```

Propositional Clausal Logic - *Semantics*: example (2)

program P

```
woman;man :- human.
human :- man.
human :- woman.
```

H1 ∨...∨ Hn ∨ ¬B1 ∨...∨ ¬Bm

for all clauses: either one atom in head is true or one atom in body is false

4 Herbrand interpretations are models for the program

~~I={woman}~~ ~~J={woman, man}~~ K={woman, man, human}

~~L={man}~~ M={man, human}

~~N={human}~~ O={woman, human} P=∅

Propositional Clausal Logic - *Semantics*: entailment

P entails C

$$P \models C$$

clause C is a **logical consequence** of program P
if every model of P is also a model of C

program P

```
woman.
woman;man :- human.
human :- man.
human :- woman.
```

```
P ⊨ human
```

models of P

```
J = {woman,man,human}

I = {woman,human}
```

intuitively preferred: doesn't assume anything to be true that doesn't *have* to be true

Propositional Clausal Logic - *Semantics*: minimal models

no subset is a model itself

could define best model to be the minimal one

BUT

```
woman;man :- human.
human.
```

has 3 models of which 2 are minimal

```
K = {woman, human}
L = {man, human}
M = {woman,man,human}
```

clauses have at most one atom in the head

A definite logic program has a unique minimal model.

Propositional Clausal Logic - *Proof Theory*: inference rules

how to check that $P \models C$ without computing all models for P and checking that each is a model for C?

by applying inference rules, C can be derived from P: $P \vdash C$

purely syntactic, not concerned with semantics

e.g., resolution

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

             has_wife;bachelor:-man,adult
```

happens to be a logical consequence of the program consisting of both input clauses

Propositional Clausal Logic - *Proof Theory*: case analysis of resolution

¬man ∨ ¬adult ∨ **married** ∨ bachelor

¬man ∨ **¬married** ∨ has_wife

either married, in order for second clause to be true as well:

¬man ∨ has_wife

or ¬married, in order for first clause to be true as well:

¬man ∨ ¬adult ∨ bachelor

therefore

¬man ∨ ¬adult ∨ bachelor ∨ ¬man ∨ has_wife

Propositional Clausal Logic - *Proof Theory*: special cases of resolution

resolution

$$\frac{\begin{array}{l} E_1 \vee E_2 \\ \neg E_2 \vee E_3 \end{array}}{E_1 \vee E_3}$$

E2 absent
E1=A
E3B

modus ponens

$$\frac{\begin{array}{l} A \\ \neg A \vee B \end{array}}{B} \qquad \frac{\begin{array}{l} A \\ A \Rightarrow B \end{array}}{B}$$

E1=¬A
E2=B
E3 absent

$$\frac{\begin{array}{l} \neg A \vee B \\ \neg B \end{array}}{\neg A} \qquad \frac{\begin{array}{l} A \Rightarrow B \\ \neg B \end{array}}{\neg A}$$

modus tollens

If it's raining it's wet; it's not wet, so it's not raining

Propositional Clausal Logic - *Proof Theory*:
successive applications of the resolution inference rule

A proof or derivation of a clause C from a program P
is a sequence of clauses $C_0,\ldots,C_n=C$
such that $\forall i_{0\ldots n}$: either $C_i \in P$ or C_i is the resolvent of C_{i1} and C_{i2} ($i_1 < i, i_2 < i$).

If there is a proof of C from P, we write $P \vdash C$

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

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

resolvent

can be
used in further
resolutions

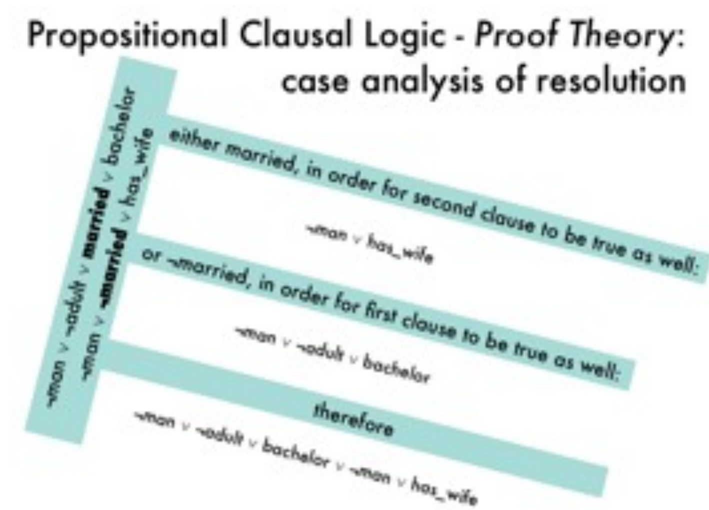
Propositional Clausal Logic - *Meta-theory*:

resolution is sound for propositional clausal logic

if $P \vdash C$ then $P \models C$

because every model of the two input clauses is also a model for the resolvent

by case analysis on truth value of resolvent



Propositional Clausal Logic - *Meta-theory*: resolution is incomplete

incomplete

the tautology `a :- a` is true under any interpretation

hence any model for a program P is also a model of `a :- a`

hence P $\models$ `a :- a`

however, resolution cannot establish P $\vdash$ `a :- a`

Propositional Clausal Logic - *Meta-theory*: resolution is refutation-complete

it derives the empty clause from any inconsistent set of clauses

entailment reformulated

P ⊨ C

⇔ each model of P is also a model of C

⇔ no model of P is a model of ¬C

⇔ P∪¬C has no model

P∪¬C is inconsistent

$C = L_1 \vee L_2 \vee \ldots \vee L_n$

$\neg C = \neg L_1 \wedge \neg L_2 \ldots \wedge \neg L_n$

$= \{\neg L_1, \neg L_2 \ldots, \neg L_n\}$

= set of clauses itself

refutation-complete

it can be shown that:

if Q is inconsistent then Q ⊢ □

if P ⊨ C then P∪¬C ⊢ □

empty clause false :- true for which no model exists

Propositional Clausal Logic - *Meta-theory*: example proof by refutation using resolution

P

```
happy :- has_friends.
friendly :- happy.
```

⊨

```
friendly :- has_friends.
```

C

P∪¬C

```
happy :- has_friends.
friendly :- happy.
has_friends.
:- friendly.
```

=¬(friendly:-has_friends)
=¬(friendly∨¬has_friends)
=¬friendly∧has_friends

P∪¬C ⊢ □