Declarative Programming

2: theoretical backgrounds
Logic Systems: structure and meta-theoretical properties

- **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.

**Soundness**: Anything you can prove is true.

**Completeness**: Anything that is true can be proven.

**Weakest form**: Prove nothing.
Logic Systems: roadmap towards Prolog

propositional clausal logic

married; bachelor :- man, adult.

relational clausal logic

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

full clausal logic

loves(X, person_loved_by(X)).

definite clause logic

no disjunction in head

lacks control constructs, arithmetic of full Prolog
Propositional Clausal Logic - Syntax: clauses

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

optional
zero or more

"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

A clause

\[ H_1; \ldots; H_n : \neg B_1, \ldots, B_m \]

is equivalent to

\[ H_1 \lor \ldots \lor H_n \lor \neg B_1 \lor \ldots \lor \neg B_m \]

hence a clause can also be defined as a disjunction of literals \( L_1 \lor L_2 \lor \ldots \lor L_n \) 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

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

(human ⇒ (woman ∨ man)) ∧ (man ⇒ human) ∧ (woman ⇒ human)

is equivalent to

(¬human ∨ woman ∨ man) ∧ (¬man ∨ human) ∧ (¬woman ∨ human)

B ⇒ H
≡ ¬B ∨ H
Propositional Clausal Logic - Syntax: special clauses

- an empty body stands for true
  - man :-.
  - true \Rightarrow \text{man}

- an empty head stands for false
  - :- impossible.
  - impossible \Rightarrow \text{false}

- man \land \neg \text{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 $P$
- mapping from Herbrand base $B_P$ to the set of truth values
  \[ i : B_P \rightarrow \{\text{true, false}\} \]

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

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 $P$

| $woman; man$ | $:- \text{human}$ |
| $human$ | $:- \text{man}$ |
| $human$ | $:- \text{woman}$ |

Herbrand base $B_P$

$\{woman, man, human\}$

$2^3$ possible Herbrand Interpretations

| $I=\{woman\}$ | $J=\{woman, man\}$ | $K=\{woman, man, human\}$ |
| $L=\{man\}$ | $M=\{man, human\}$ |
| $N=\{human\}$ | $O=\{woman, human\}$ |

$n=\{(woman, false), (man, false), (human, false)\}$

$P=\emptyset$
Propositional Clausal Logic - Semantics: example (2)

program P

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

4 Herbrand interpretations are models for the program

I=\{woman\}  J=\{woman, man\}  K=\{woman, man, human\}
L=\{man\}    M=\{man, human\}   O=\{woman, human\}
N=\{human\}  P=\Ø

for all clauses: either one atom in head is true or one atom in body is false
Propositional Clausal Logic - Semantics: entailment

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

program P

\[
\begin{align*}
\text{woman}. \\
\text{woman;man} & : - \text{human}. \\
\text{human} & : - \text{man}. \\
\text{human} & : - \text{woman}. \\
\end{align*}
\]

models of P

\[
\begin{align*}
J & = \{\text{woman, man, human}\} \\
I & = \{\text{woman, human}\} \\

\text{intuitively preferred: doesn't assume anything to be true that doesn't have to be true}
\end{align*}
\]
Propositional Clausal Logic - Semantics: minimal models

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}

A definite logic program has a unique minimal model.
Propositional Clausal Logic - Proof Theory: inference rules

how to check that \( P \not\models C \) 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

\[
\begin{align*}
\text{has\_wife} & : -\text{man,married} \\
\text{married} & ; \text{bachelor} : -\text{man,adult} \\
\text{has\_wife} ; \text{bachelor} & : -\text{man,adult}
\end{align*}
\]

happens to be a logical consequence of the program consisting of both input clauses
Propositional Clausal Logic - Proof Theory: case analysis of resolution

either married, in order for second clause to be true as well:
\[ \neg \text{man} \lor \text{has\_wife} \]

or \neg\text{married}, in order for first clause to be true as well:
\[ \neg \text{man} \lor \neg \text{adult} \lor \text{bachelor} \]

therefore
\[ \neg \text{man} \lor \neg \text{adult} \lor \text{bachelor} \lor \neg \text{man} \lor \text{has\_wife} \]
Propositional Clausal Logic - Proof Theory: special cases of resolution

E₁ ∨ E₂
¬E₂ ∨ E₃
E₁ ∨ E₃

E₂ absent
E₁=A
E₃B

E₁=¬A
E₂=B
E₃ absent

modus ponens

A
¬A ∨ B
B
A
A ⇒ B
B

modus tollens

¬A ∨ B
¬B
A
A ⇒ B
¬B
¬A

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 $C_{i_1}$ and $C_{i_2}$ ($i_1 < i, i_2 < i$).

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

\[\text{square} : - \text{rectangle}, \text{equal}_\text{sides} \quad \text{rectangle} : - \text{parallelogram}, \text{right}_\text{angles}\]

\[\text{square} : - \text{parallelogram}, \text{right}_\text{angles}, \text{equal}_\text{sides}\]

(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

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 $P \models a :- a$
Propositional Clausal Logic - Meta-theory: resolution is refutation-complete

*P ⊨ C*

⇔ each model of P is also a model of C
⇔ no model of P is a model of ¬C
⇔ \(P \cup \neg C\) has no model
⇔ \(P \cup \neg C\) is inconsistent

\[C = L_1 \lor L_2 \lor \ldots \lor L_n\]
\[\neg C = \neg L_1 \land \neg L_2 \land \ldots \land \neg L_n\]
\[= \{\neg L_1, \neg L_2, \ldots, \neg L_n\}\]
\[= \text{set of clauses itself}\]

it can be shown that:
if Q is inconsistent then \(Q \vdash \Box\)
if \(P \vdash C\) then \(P \cup \neg C \vdash \Box\)

empty clause false :- true for which no model exists

it derives the empty clause from any inconsistent set of clauses
Propositional Clausal Logic - Meta-theory: example proof by refutation using resolution

\[
P \cup \neg C \vdash \square
given:
\begin{align*}
happy & :- has\_friends. \\
friendly & :- happy. \\
\end{align*}
\[
P \cup \neg C
\begin{align*}
happy & :- has\_friends. \\
friendly & :- happy. \\
has\_friends. & :- friendly. \\
\end{align*}
\[
\neg (\neg \text{friendly} \lor \neg \text{has\_friends}) \\
\neg \text{friendly} \land \text{has\_friends}
\]

\[
P \cup \neg C \vdash \square
\begin{align*}
:- \text{friendly} & \\
\text{friendly} & :- \neg \text{happy} \\
:- \text{happy} & \\
\text{happy} & :- \neg \text{has\_friends} \\
:- \text{has\_friends} & \\
\text{has\_friends} & \\
\end{align*}
\]