Often you want to assume that your knowledge is complete.

**Example:** assume that a database of what students are enrolled in a course is complete. We don’t want to have to state all negative enrolment facts!

The definite clause language is **monotonic:** adding clauses can’t invalidate a previous conclusion.

Under the complete knowledge assumption, the system is **non-monotonic:** adding clauses can invalidate a previous conclusion.
Equality

Equality is a special predicate symbol with a standard domain-independent intended interpretation.

- Suppose interpretation $I = \langle D, \phi, \pi \rangle$.
- $t_1$ and $t_2$ are ground terms then $t_1 = t_2$ is true in interpretation $I$ if $t_1$ and $t_2$ denote the same individual. That is, $t_1 = t_2$ if $\phi(t_1)$ is the same as $\phi(t_2)$.
- $t_1 \neq t_2$ when $t_1$ and $t_2$ denote different individuals.

Example:

$D = \{c, d, e\}$.

$\phi(\text{phone}) = a$,
$\phi(\text{pencil}) = b$,
$\phi(\text{telephone}) = c$.

What equalities and inequalities hold?

- $\text{phone} = \text{telephone}$,
- $\text{phone} = \text{phone}$,
- $\text{pencil} = \text{pencil}$,
- $\text{telephone} = \text{telephone}$,
- $\text{pencil} \neq \text{phone}$,
- $\text{pencil} \neq \text{telephone}$.

Equality does not mean similarity!
Equality is a special predicate symbol with a standard domain-independent intended interpretation.

- Suppose interpretation $I = \langle D, \phi, \pi \rangle$.
- $t_1$ and $t_2$ are ground terms then $t_1 = t_2$ is true in interpretation $I$ if $t_1$ and $t_2$ denote the same individual. That is, $t_1 = t_2$ if $\phi(t_1)$ is the same as $\phi(t_2)$.
- $t_1 \neq t_2$ when $t_1$ and $t_2$ denote different individuals.

Example:

$D = \{\text{phone}, \text{pencil}, \text{telephone} \}$.

$\phi(\text{phone}) = \text{phone}, \phi(\text{pencil}) = \text{pencil}, \phi(\text{telephone}) = \text{telephone}$

What equalities and inequalities hold?
Equality is a special predicate symbol with a standard domain-independent intended interpretation.

- Suppose interpretation $I = \langle D, \phi, \pi \rangle$.

- $t_1$ and $t_2$ are ground terms then $t_1 = t_2$ is true in interpretation $I$ if $t_1$ and $t_2$ denote the same individual. That is, $t_1 = t_2$ if $\phi(t_1)$ is the same as $\phi(t_2)$.

- $t_1 \neq t_2$ when $t_1$ and $t_2$ denote different individuals.

- Example:
  
  $D = \{\text{phone}, \text{pencil}, \text{telephone}\}$.
  
  $\phi(\text{phone}) = \text{phone}$, $\phi(\text{pencil}) = \text{pencil}$, $\phi(\text{telephone}) = \text{telephone}$

  What equalities and inequalities hold?
  
  $\text{phone} = \text{telephone}$, $\text{phone} = \text{phone}$, $\text{pencil} = \text{pencil}$, $\text{telephone} = \text{telephone}$
  
  $\text{pencil} \neq \text{phone}$, $\text{pencil} \neq \text{telephone}$

Equality does not mean similarity!
Equality

Equality is a special predicate symbol with a standard domain-independent intended interpretation.

- Suppose interpretation $I = \langle D, \phi, \pi \rangle$.
- $t_1$ and $t_2$ are ground terms then $t_1 = t_2$ is true in interpretation $I$ if $t_1$ and $t_2$ denote the same individual. That is, $t_1 = t_2$ if $\phi(t_1)$ is the same as $\phi(t_2)$.
- $t_1 \neq t_2$ when $t_1$ and $t_2$ denote different individuals.
- Example:
  $D = \{\text{phone}, \text{pencil}, \text{telephone}\}$.
  $\phi(\text{phone}) = \text{phone}, \phi(\text{pencil}) = \text{pencil}, \phi(\text{telephone}) = \text{telephone}$
  What equalities and inequalities hold?
  $\text{phone} = \text{telephone}, \text{phone} = \text{phone}, \text{pencil} = \text{pencil}, \text{telephone} = \text{telephone}$
  $\text{pencil} \neq \text{phone}, \text{pencil} \neq \text{telephone}$
- Equality does not mean similarity!
Equality is:

- **Reflexive**: $X = X$
- **Symmetric**: if $X = Y$ then $Y = X$
- **Transitive**: if $X = Y$ and $Y = Z$ then $X = Z$

For each $n$-ary function symbol $f$

$$f(X_1, \ldots, X_n) = f(Y_1, \ldots, Y_n) \text{ if } X_1 = Y_1 \text{ and } \cdots \text{ and } X_n = Y_n.$$  

For each $n$-ary predicate symbol $p$

$$p(X_1, \ldots, X_n) \text{ if } p(Y_1, \ldots, Y_n) \text{ and } X_1 = Y_1 \text{ and } \cdots \text{ and } X_n = Y_n.$$
Suppose the only clauses for enrolled are:

- \textit{enrolled}((\textit{sam}, \textit{cs222}))
- \textit{enrolled}((\textit{chris}, \textit{cs222}))
- \textit{enrolled}((\textit{sam}, \textit{cs873}))

To conclude \textit{\neg enrolled}((\textit{chris}, \textit{cs873})), what do we need to assume?
Unique Names Assumption

- Suppose the only clauses for enrolled are
  
  \[
  \text{enrolled}(\text{sam}, \text{cs222})
  \]
  
  \[
  \text{enrolled}(\text{chris}, \text{cs222})
  \]
  
  \[
  \text{enrolled}(\text{sam}, \text{cs873})
  \]

To conclude \(\neg \text{enrolled}(\text{chris}, \text{cs873})\), what do we need to assume?

- All other enrolled facts are false
- Inequalities:

\[
\text{sam} \neq \text{chris} \land \text{cs873} \neq \text{cs222}
\]

- The unique names assumption (UNA) is the assumption that distinct ground terms denote different individuals.
Suppose the rules for atom $a$ are

\[
a \leftarrow b_1.
\]

\[
\vdots
\]

\[
a \leftarrow b_n.
\]

equivalently $a \leftarrow b_1 \lor \ldots \lor b_n$.

Under the Complete Knowledge Assumption, if $a$ is true, one of the $b_i$ must be true:

\[
a \rightarrow b_1 \lor \ldots \lor b_n.
\]

Thus, the clauses for $a$ mean

\[
a \leftrightarrow b_1 \lor \ldots \lor b_n
\]
Clark Normal Form

The Clark normal form of the clause

\[ p(t_1, \ldots, t_k) \leftarrow B. \]

is the clause

\[ p(V_1, \ldots, V_k) \leftarrow \exists W_1 \ldots \exists W_m \ V_1 = t_1 \land \ldots \land V_k = t_k \land B. \]

where

- \( V_1, \ldots, V_k \) are \( k \) variables that did not appear in the original clause
- \( W_1, \ldots, W_m \) are the original variables in the clause.
Clark Normal Form

The Clark normal form of the clause

\[ p(t_1, \ldots, t_k) \leftarrow B. \]

is the clause

\[ p(V_1, \ldots, V_k) \leftarrow \exists W_1 \ldots \exists W_m \; V_1 = t_1 \land \ldots \land V_k = t_k \land B. \]

where

- \( V_1, \ldots, V_k \) are \( k \) variables that did not appear in the original clause
- \( W_1, \ldots, W_m \) are the original variables in the clause.
- When the clause is an atomic clause, \( B \) is true.
- Often can be simplified by replacing \( \exists W \; V = W \land p(W) \) with \( P(V) \).
Clark normal form

For the clauses

\[ \text{student}(\text{mary}). \]
\[ \text{student}(\text{sam}). \]
\[ \text{student}(X) \leftarrow \text{undergrad}(X). \]

the Clark normal form is

\[ \text{student}(V) \leftarrow V = \text{mary}. \]
\[ \text{student}(V) \leftarrow V = \text{sam}. \]
\[ \text{student}(V) \leftarrow \exists X \ V = X \land \text{undergrad}(X). \]
Clark’s Completion

Suppose all of the clauses for $p$ are put into Clark normal form, with the same set of introduced variables, giving

$$p(V_1, \ldots, V_k) \leftarrow B_1.$$  

$$\vdots$$  

$$p(V_1, \ldots, V_k) \leftarrow B_n.$$  

which is equivalent to

$$p(V_1, \ldots, V_k) \leftarrow B_1 \lor \ldots \lor B_n.$$  

Clark’s completion of predicate $p$ is the equivalence

$$\forall V_1 \ldots \forall V_k p(V_1, \ldots, V_k) \leftrightarrow B_1 \lor \ldots \lor B_n$$

If there are no clauses for $p$,  

\[ \]
Clark’s Completion

Suppose all of the clauses for $p$ are put into Clark normal form, with the same set of introduced variables, giving

$$p(V_1, \ldots, V_k) \leftarrow B_1.$$  

\[ \vdots \]  

$$p(V_1, \ldots, V_k) \leftarrow B_n.$$  

which is equivalent to

$$p(V_1, \ldots, V_k) \leftarrow B_1 \lor \ldots \lor B_n.$$  

Clark’s completion of predicate $p$ is the equivalence

$$\forall V_1 \ldots \forall V_k \ p(V_1, \ldots, V_k) \leftrightarrow B_1 \lor \ldots \lor B_n$$

If there are no clauses for $p$, the completion results in

$$\forall V_1 \ldots \forall V_k \ p(V_1, \ldots, V_k) \leftrightarrow false$$

Clark’s completion of a knowledge base consists of the completion of every predicate symbol along the unique names assumption.
Completion example

\[ p \leftarrow q \land \neg r. \]
\[ p \leftarrow s. \]
\[ q \leftarrow \neg s. \]
\[ r \leftarrow \neg t. \]
\[ t. \]
\[ s \leftarrow w. \]
Completion Example

Consider the recursive definition:

\[
passed\_each([], St, MinPass).
\]

\[
passed\_each([C|R], St, MinPass) \leftarrow
\]

\[
passed(St, C, MinPass) \land
\]

\[
passed\_each(R, St, MinPass).
\]

In Clark normal form, this can be written as
Completion Example

Consider the recursive definition:

\[
\text{passed}\text{\_each}([], St, \text{MinPass}).
\]
\[
\text{passed}\text{\_each}([C|R], St, \text{MinPass}) \leftarrow \\
\quad \text{passed}(St, C, \text{MinPass}) \land \\
\quad \text{passed}\text{\_each}(R, St, \text{MinPass}).
\]

In Clark normal form, this can be written as

\[
\text{passed}\text{\_each}(L, S, M) \leftarrow L = [].
\]
\[
\text{passed}\text{\_each}(L, S, M) \leftarrow \\
\quad \exists C \exists R L = [C|R] \land \text{passed}(S, C, M) \land \text{passed}\text{\_each}(R, S, M).
\]

Here we renamed the variables as appropriate. Thus, Clark’s completion of \text{passed}\text{\_each} is
Completion Example

Consider the recursive definition:

\[
\text{passed\_each}([], St, \text{MinPass}).
\]
\[
\text{passed\_each}([C \mid R], St, \text{MinPass}) \leftarrow
\]
\[
\text{passed}(St, C, \text{MinPass}) \land
\]
\[
\text{passed\_each}(R, St, \text{MinPass}).
\]

In Clark normal form, this can be written as

\[
\text{passed\_each}(L, S, M) \leftarrow L = [].
\]
\[
\text{passed\_each}(L, S, M) \leftarrow
\]
\[
\exists C \exists R L = [C \mid R] \land \text{passed}(S, C, M) \land \text{passed\_each}(R, S, M).
\]

Here we renamed the variables as appropriate. Thus, Clark’s completion of \text{passed\_each} is

\[
\forall L \forall S \forall M \text{passed\_each}(L, S, M) \leftrightarrow L = [] \lor
\]
\[
\exists C \exists R L = [C \mid R] \land \text{passed}(S, C, M) \land \text{passed\_each}(R, S, M).
\]
Clark’s completion of a knowledge base consists of the completion of every predicate.

The completion of an $n$-ary predicate $p$ with no clauses is $p(V_1, \ldots, V_n) \Leftrightarrow \text{false}$.

You can interpret negations in the body of clauses. $\sim a$ means $a$ is false under the complete knowledge assumption. $\sim a$ is replaced by $\neg a$ in the completion. This is negation as failure.
Defining *empty_course*

Given database of:

- \( \text{course}(C) \) that is true if \( C \) is a course
- \( \text{enrolled}(S, C) \) that is true if student \( S \) is enrolled in course \( C \).

Define *empty_course*(\( C \)) that is true if there are no students enrolled in course \( C \).
Defining *empty_course*

Given database of:

- \(\text{course}(C)\) that is true if \(C\) is a course
- \(\text{enrolled}(S, C)\) that is true if student \(S\) is enrolled in course \(C\).

Define *empty_course*(\(C\)) that is true if there are no students enrolled in course \(C\).

- Using negation as failure, *empty_course*(\(C\)) can be defined by
  \[
  \text{empty}_\text{course}(C) \leftarrow \text{course}(C) \land \neg \text{has_enrollment}(C).
  \]
  \[
  \text{has_enrollment}(C) \leftarrow \text{enrolled}(S, C).
  \]
Defining `empty_course`

Given database of:

- `course(C)` that is true if `C` is a course
- `enrolled(S, C)` that is true if student `S` is enrolled in course `C`.

Define `empty_course(C)` that is true if there are no students enrolled in course `C`.

- Using negation as failure, `empty_course(C)` can be defined by

  \[
  \text{empty}_{\text{course}}(C) \leftarrow \text{course}(C) \land \neg \text{has}_{\text{enrollment}}(C).
  \]

  \[
  \text{has}_{\text{enrollment}}(C) \leftarrow \text{enrolled}(S, C).
  \]

- The completion of this is:
Defining \textit{empty\_course}

Given database of:

- \textit{course}(C) that is true if \( C \) is a course
- \textit{enrolled}(S, C) that is true if student \( S \) is enrolled in course \( C \).

Define \textit{empty\_course}(C) that is true if there are no students enrolled in course \( C \).

Using negation as failure, \textit{empty\_course}(C) can be defined by

\[
\text{empty\_course}(C) \leftarrow \text{course}(C) \land \lnot \text{has\_enrollment}(C).
\]
\[
\text{has\_enrollment}(C) \leftarrow \text{enrolled}(S, C).
\]

The completion of this is:

\[
\forall C \text{ empty\_course}(C) \iff \text{course}(C) \land \lnot \text{has\_enrollment}(C).
\]
\[
\forall C \text{ has\_enrollment}(C) \iff \exists S \text{ enrolled}(S, C).
\]
Bottom-up negation as failure interpreter

\[ C := \{\}; \]
repeat
    either
        select \( r \in KB \) such that
            \( r \) is “\( h \leftarrow b_1 \land \ldots \land b_m \)”
            \( b_i \in C \) for all \( i \), and
            \( h \notin C \);
        \( C := C \cup \{h\} \)
    or
        select \( h \) such that for every rule “\( h \leftarrow b_1 \land \ldots \land b_m \)” \( \in KB \)
            either for some \( b_i, \sim b_i \in C \)
            or some \( b_i = \sim g \) and \( g \in C \)
        \( C := C \cup \{\sim h\} \)
    until no more selections are possible
Negation as failure example

\[ p \leftarrow q \land \sim r. \]
\[ p \leftarrow s. \]
\[ q \leftarrow \sim s. \]
\[ r \leftarrow \sim t. \]
\[ t. \]
\[ s \leftarrow w. \]
If the proof for $a$ fails, you can conclude $\sim a$.

Failure can be defined recursively: Suppose you have rules for atom $a$:

$$a \leftarrow b_1$$
$$\vdots$$
$$a \leftarrow b_n$$

If each body $b_i$ fails, $a$ fails.

A body fails if one of the conjuncts in the body fails.

Note that you need *finite* failure. Example $p \leftarrow p$. 
Floundering

\[ p(X) \leftarrow \neg q(X) \land r(X). \]
\[ q(a). \]
\[ q(b). \]
\[ r(d). \]
\[ \text{ask } p(X). \]

- What is the answer to the query?
Floundering

\[ p(X) \leftarrow \sim q(X) \land r(X). \]
\[ q(a). \]
\[ q(b). \]
\[ r(d). \]
\[ \text{ask } p(X). \]

- What is the answer to the query?
- How can a top-down proof procedure find the answer?
Floundering

\[ p(X) \leftarrow \neg q(X) \land r(X). \]

\[ q(a). \]

\[ q(b). \]

\[ r(d). \]

\[ \text{ask } p(X). \]

- What is the answer to the query?
- How can a top-down proof procedure find the answer?
- Delay the subgoal until it is bound enough.
  Sometimes it never gets bound enough — “floundering”.
Problematic Cases

\[ p(X) \leftarrow \sim q(X) \]
\[ q(X) \leftarrow \sim r(X) \]
\[ r(a) \]
ask \( p(X) \).

What is the answer?
Problematic Cases

\[ p(X) \leftarrow \sim q(X) \]
\[ q(X) \leftarrow \sim r(X) \]
\[ r(a) \]
\[ \text{ask } p(X). \]

- What is the answer?
- What does delaying do?
Problematic Cases

\[ p(X) \leftarrow \sim q(X) \]
\[ q(X) \leftarrow \sim r(X) \]
\[ r(a) \]
\[ \text{ask } p(X). \]

- What is the answer?
- What does delaying do?
- How can this be implemented?