Logic Programming

Theory Lecture 7:
The Closed World Assumption

Alex Simpson

School of Informatics

8th November 2012
Negation as failure: problem 1

Prolog’s treatment of *negation as failure* is a procedural rather than declarative operation.

For example, the program

```
woman(alice).
man(bob).
```

gives rise to the following queries and responses

```
?- \+ man(X), woman(X).
no
```

```
?- woman(X), \+ man(X).
X = alice
```

So the comma “,” can no longer be understood as the commutative operation of logical conjunction.
Negation as failure: problem 1

Prolog’s treatment of negation as failure is a procedural rather than declarative operation.

For example, the program

    woman(alice).
    man(bob).

gives rise to the following queries and responses

?- \+ man(X), woman(X).
no

?- woman(X), \+ man(X).
X = alice

So the comma “,” can no longer be understood as the commutative operation of logical conjunction.

The problem here is the use of negation on non-ground formulas.
Negation as failure: problem 2

Even for ground instances of negation, the procedural behaviour or Prolog programs does not match the declarative reading.

Example program and query.

```
p :- \+ p.

?- p.
```

Prolog goes into a loop on this, but under the declarative reading, \( p \) is a logical consequence of the theory (this was covered in Question 3 of Tutorial 3).
Negation as failure: problem 2

Even for ground instances of negation, the procedural behaviour or Prolog programs does not match the declarative reading.

Example program and query.

\[ p :- \neg p. \]

?- p.

Prolog goes into a loop on this, but under the declarative reading, \( p \) is a logical consequence of the theory (this was covered in Question 3 of Tutorial 3).

The problem here is the inclusion of negated formulas in the program.
Restricting negation as failure

Today we shall make declarative sense of negation as failure with two major restrictions.

- Only ground atomic formulas appear negated.
- Negations appear only in queries, not in programs.
Even this restrictive setting causes complications.

\[ \text{woman(alice).} \]
\[ \text{man(bob).} \]

\[ \text{?- \( \neg \) man(alice).} \]
\[ \text{yes} \]

However,

\[ \text{woman(alice), man(bob) \not\models \neg \text{man(alice)}} \]

Thus the conclusion is not a logical consequence of the program.
Even this restrictive setting causes complications.

\[
\text{woman(alice).}
\]
\[
\text{man(bob).}
\]

\[
?- \ 
+\text{man(alice)}.
\]
\[
\text{yes}
\]

However,

\[
\text{woman(alice), man(bob) } \not\models \neg \text{man(alice)}
\]

Thus the conclusion is not a logical consequence of the program.

We address this by “completing” the program to a larger logical theory of which the conclusion is a consequence.
Circumscribing knowledge

We view our program as a logical theory expressing knowledge about the world.

In several situations, it is convenient to assume that the program contains complete information about certain kinds of logical statements.

We can then make additional inferences about the world based on the assumed completeness of our knowledge.

The *Closed World Assumption (CWA)* makes the assumption that the program contains complete knowledge about which ground atomic formulas are true.
Consider our simple example

woman(alice).
man(bob).

If we impose the CWA then we can conclude that neither woman(bob) nor man(alice) hold, since if either of these were true then our program would not contain complete knowledge about true ground atomic formulas.

Thus the CWA allows us to “complete” our knowledge base to the larger theory

woman(alice), man(bob), ¬woman(bob), ¬man(alice)

Note that we are here adding negated atomic formulas to the knowledge base.
Closed World Assumption in general

A theory $T$ in predicate logic is a set of sentences. (We do not insist that $T$ contain only definite clauses.)

A theory $T$ is said to be complete for ground atomic formulas if, for every ground atomic formula $A$, either $T \models A$ or $T \models \neg A$.

Let $T$ be any theory (it does not have to be complete for ground atomic formulas). We define a new theory $CWA(T)$ by:

$$CWA(T) = T \cup \{ \neg A \mid A \text{ is a ground atomic formula and } T \not\models A \}$$

By definition, the theory $CWA(T)$ is complete for ground atomic formulas.

(Formulas that are either atomic or negated atomic are often called literals. Thus $CWA(T)$ is complete for ground literals.)
Example

Let $T$ be the theory (in definite clause logic)

$$\forall X. \text{cheap}(X) \rightarrow \text{nasty}(X)$$
$$\forall X. \text{free}(X) \rightarrow \text{cool}(X)$$

cheap(windows)
free(linux)
cool(mac)
Example

Let $T$ be the theory (in definite clause logic)

$\forall X. \text{cheap}(X) \rightarrow \text{nasty}(X)$
$\forall X. \text{free}(X) \rightarrow \text{cool}(X)$

cheap(windows),
free(linux),
cool(mac)

Then $CWA(T)$ adds the following ground atomic formulas to $T$

$\neg\text{free}(\text{windows}), \neg\text{cool}(\text{windows}), \neg\text{cheap}(\text{linux}),$
$\neg\text{nasty}(\text{linux}), \neg\text{free}(\text{mac}), \neg\text{cheap}(\text{mac}), \neg\text{nasty}(\text{mac})$
Closed World Assumption in definite clause logic

Note that the theory $CWA(T)$ is not itself a definite clause theory, even when $T$ is a definite clause theory. However, the assumption that $T$ is a definite clause theory has one important consequence.

**Theorem**
Let $T$ be a set of definite clauses. Then the theory $CWA(T)$ is consistent (that is, there is no formula $F$ such that $CWA(T) \models F$ and $CWA(T) \models \neg F$).

**Proof**
For a ground atomic formula $A$, the minimal Herbrand model $\mathcal{H}$ of $T$ satisfies that $\mathcal{H} \models A$ if and only if $T \models A$ (see Lecture 6 “Importance of minimal Herbrand model”). Hence, if $T \not\models A$ then $\mathcal{H} \models \neg A$. Thus the minimal Herbrand model for $T$ is also a model for $CWA(T)$. Any theory that has a model is consistent.
Soundness of negated queries

Suppose $T$ is a definite clause theory and $A$ is a ground atomic formula. Suppose also that the prolog query

$$\texttt{?- \lnot + A}.$$ 

succeeds. Then $CWA(T) \models \lnot A$. Also $\mathcal{H} \models \lnot A$, where $\mathcal{H}$ is the minimal Herbrand model for $T$.

Prolog’s behaviour on queries including negated ground queries can thus be considered either as sound relative to the Closed World Assumption, or as sound relative to the minimal Herbrand model.
Non-monotonicity

Logical consequence is *monotonic* in the sense that, given two theories $T, T'$ with $T \subseteq T'$ then it holds that

$$T \models F \quad \text{implies} \quad T' \models F,$$

for all formulas $F$.

The Closed World Assumption is *non-monotonic* in the sense that, given two theories $T, T'$ with $T \subseteq T'$ then it *does not* hold in general that

$$\text{CWA}(T) \models F \quad \text{implies} \quad \text{CWA}(T') \models F,$$

for all formulas $F$. 
Example (continued)

Let $T'$ be our example theory extended with the new axiom

$$\text{cheap(linux)}$$

Then $CWA(T')$ adds the following ground atomic formulas to $T'$

$$\neg\text{free(windows)}, \neg\text{cool(windows)},$$
$$\neg\text{free(mac)}, \neg\text{cheap(mac)}, \neg\text{nasty(mac)}$$

We have $T \subseteq T'$ and

$$CWA(T) \models \neg\text{cheap(linux)} \text{ but } CWA(T') \not\models \neg\text{cheap(linux)}$$

where the latter claim holds because $CWA(T') \models \text{cheap(linux)}$ and $CWA(T')$ is consistent (since $T'$ is a definite clause theory).

This illustrates non-monotonicity.
Theorem. Let $T$ be a definite clause theory, and let $H$ be its minimum Herbrand model.

Then the statements below, about any ground atomic formula $A$,

1. The Prolog query $\vdash A$ succeeds.
2. $\text{CWA}(T) \models \neg A$.
3. $H \models \neg A$.

enjoy the following implications

$$1 \implies 2 \iff 3$$
Two issues with the CWA

1. Because of the *undecidability* of definite clause predicate logic (see Lecture 5), it is not possible to effectively compute the theory $CWA(T)$ from the theory $T$.

2. The CWA over-approximates the behaviour of negation by failure. Consider the propositional theory $T$ consisting of a single axiom

$$p \rightarrow p$$

Then the Prolog query \+ $p$ goes into a loop. Nevertheless,

$$CWA(T) \models \neg p$$
Main points today

Procedural nature of negation by failure

Closed World Assumption

Non-monotonnicity of CWA