Logic Programming

Theory Lecture 8: Clark Completion

Alex Simpson

School of Informatics

12th November 2012

Two issues with the CWA (Recap Lecture 7)

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

   $$\mathtt{p} \rightarrow \mathtt{p}$$

   Then the Prolog query $\backslash+\,\mathtt{p}$ goes into a loop. Nevertheless,

   $$CWA(T) \models \neg\mathtt{p}$$

Clark completion

The *Clark completion* is an alternative completion procedure, used for modelling negation by failure.

In contrast to the CWA, the Clark completion *is* effectively computable.

However, whereas $CWA(T)$ can be defined for any logical theory T, the Clark completion requires T to be in a particular form (see next slide).

Roughly, the Clark completion makes the assumption that the axioms of the definite clause program completely axiomatize all possible reasons for atomic formulas to be true.

NAF logic program

A *negation as failure (NAF)* logic program is a collection of formulas of the two shapes.

$$B$$

$$L_1 \wedge \cdots \wedge L_k \rightarrow B$$

where B is an *atomic formula* and $L_1, \ldots, L_k$ are *literals* (atomic formulas or their negations).

A *NAF goal* is a list $G_1, \ldots, G_m$ of *literals*.

The Clark completion can be defined generally for NAF logic programs and goals.

However, for simplicity, we make the same restrictions as in Lecture 8: we allow negations only in queries, and we only allow ground atomic formulas to be negated.

Example

Suppose we have a theory in which the only clauses with *head predicate* `british` are

$$\begin{aligned} \forall \texttt{X}.\ \texttt{english(X)} &\rightarrow \texttt{british(X)} \\ \forall \texttt{X}.\ \texttt{scottish(X)} &\rightarrow \texttt{british(X)} \\ \forall \texttt{X}.\ \texttt{welsh(X)} &\rightarrow \texttt{british(X)} \end{aligned}$$

Then the Clark formula for the predicate `british` is

$$\forall \texttt{X}.(\texttt{british(X)} \leftrightarrow (\texttt{english(X)} \vee \texttt{scottish(X)} \vee \texttt{welsh(X)}))$$

(The *head predicate* in a clause is: the predicate on the right-hand side of the implication, if the clause is an implication; and the only predicate in the formula, if the clause is an atomic formula.)

Example (continued)

Suppose the remaining axioms are

$$\texttt{english(elizabeth)}$$
$$\texttt{scottish(mary)}$$
$$\texttt{scottish(james)}$$

Then the Clark formulas for the three predicates `english`, `scottish`, `welsh` are

$$\forall \texttt{X}.\,(\texttt{english(X)} \leftrightarrow \texttt{X = elizabeth})$$
$$\forall \texttt{X}.\,(\texttt{scottish(X)} \leftrightarrow (\texttt{X = mary} \vee \texttt{X = james}))$$
$$\forall \texttt{X}.\,(\neg\texttt{welsh(X)})$$

Example 2 (completed)

The Clark completion of the full theory:

$$\forall \mathtt{X}.\ \mathtt{english(X)} \rightarrow \mathtt{british(X)}\ ,$$
$$\forall \mathtt{X}.\ \mathtt{scottish(X)} \rightarrow \mathtt{british(X)}\ ,$$
$$\forall \mathtt{X}.\ \mathtt{welsh(X)} \rightarrow \mathtt{british(X)}\ ,$$
$$\mathtt{english(elizabeth)}, \mathtt{scottish(mary)}, \mathtt{scottish(james)}$$

is the theory:

$$\forall \mathtt{X}.(\mathtt{british(X)} \leftrightarrow (\mathtt{english(X)} \vee \mathtt{scottish(X)} \vee \mathtt{welsh(X)}))$$
$$\forall \mathtt{X}.(\mathtt{english(X)} \leftrightarrow \mathtt{X} = \mathtt{elizabeth})$$
$$\forall \mathtt{X}.(\mathtt{scottish(X)} \leftrightarrow (\mathtt{X} = \mathtt{mary} \vee \mathtt{X} = \mathtt{james}))$$
$$\forall \mathtt{X}.(\neg \mathtt{welsh(X)})$$
$$\mathtt{elizabeth} \neq \mathtt{james} \quad \mathtt{james} \neq \mathtt{mary} \quad \mathtt{mary} \neq \mathtt{elizabeth}$$

Note that the Clark completion is not itself a definite clause theory.

General completion procedure: Step 1

We now condsider the general procedure for constructing the Clark completion $Comp(T)$ of a definite clause theory T.

First we rewrite each individual definite clause in the theory. The general form of a definite clause is

$$\forall \vec{\mathtt{X}}.\ (A_1 \wedge \cdots \wedge A_k \ \rightarrow \ p(\vec{t}))$$

where $\vec{\mathtt{X}}$ is a tuple of variables, and $\vec{t}$ is a tuple of n-terms, where n is the arity (= number of arguments) of the predicate p.

We rewrite the clause to the equivalent formula

$$\forall \vec{\mathtt{Y}}.\ (\exists \vec{\mathtt{X}}.\ A_1 \wedge \cdots \wedge A_k \wedge \vec{\mathtt{Y}} = \vec{t}) \ \rightarrow \ p(\vec{\mathtt{Y}})$$

where $\vec{\mathtt{Y}}$ is a tuple of n new variables.

Justifying this equivalence

The formula

$$\forall \vec{\mathtt{X}}.\ (A_1 \wedge \cdots \wedge A_k \ \rightarrow\ p(\vec{t}))$$

is equivalent to

$$\forall \vec{\mathtt{Y}},\ \vec{\mathtt{X}}.\ (A_1 \wedge \cdots \wedge A_k \wedge \vec{\mathtt{Y}} = \vec{t} \ \rightarrow\ p(\vec{\mathtt{Y}}))$$

which is, in turn, equivalent to the desired formula

$$\forall \vec{\mathtt{Y}}.\ (\exists \vec{\mathtt{X}}.\ A_1 \wedge \cdots \wedge A_k \wedge \vec{\mathtt{Y}} = \vec{t}) \ \rightarrow\ p(\vec{\mathtt{Y}})$$

because of the general logical equivalence

$$(\forall \mathtt{X}.\ (F \rightarrow G)) \quad \leftrightarrow \quad ((\exists \mathtt{X}.\ F) \rightarrow G)\ ,$$

which holds whenever the variable $\mathtt{X}$ does not appear in the formula G.

Some special cases

$$\forall \vec{\mathtt{X}}.\ (A_1 \wedge \cdots \wedge A_k \ \rightarrow\ p(\vec{t}))$$

For special cases of this formula, one can simplify the equivalent formulas.

When $k = 0$ (i.e., the axiom is a Prolog fact rather than rule) the equivalent formula is

$$\forall \vec{\mathtt{Y}}.\ (\exists \vec{\mathtt{X}}.\ \vec{\mathtt{Y}} = \vec{t}) \ \rightarrow\ p(\vec{\mathtt{Y}})$$

When the formula is ground, the equivalent formula simplifies to

$$\forall \vec{\mathtt{Y}}.\ A_1 \wedge \cdots \wedge A_k \wedge \vec{\mathtt{Y}} = \vec{t} \ \rightarrow\ p(\vec{\mathtt{Y}})$$

When $\vec{t}$ is the vector of variables $\vec{\mathtt{X}}$, there is no need to further rewrite the original clause

$$\forall \vec{\mathtt{X}}.\ (A_1 \wedge \cdots \wedge A_k \ \rightarrow\ p(\vec{\mathtt{X}}))$$

General completion procedure: Step 2

We have now rewritten each clause with head predicate p to an equivalent formula

$$\forall \vec{Y}.\ E \ \rightarrow\ p(\vec{Y})$$

Suppose there are m such clauses for p, giving

$$\begin{array}{rcl} \forall \vec{Y}.\ E_1 & \rightarrow & p(\vec{Y}) \\ \forall \vec{Y}.\ E_2 & \rightarrow & p(\vec{Y}) \\ & \vdots & \\ \forall \vec{Y}.\ E_m & \rightarrow & p(\vec{Y}) \end{array}$$

Taken together, these formulas are equivalent to the single formula

$$\forall \vec{Y}.\ (E_1 \vee E_2 \vee \cdots \vee E_m)\ \rightarrow\ p(\vec{Y})$$

The *Clark formula* for the predicate p is then the formula

$$\forall \vec{Y}.\ p(\vec{Y})\ \leftrightarrow\ (E_1 \vee E_2 \vee \cdots \vee E_m)$$

A special case

In the case that $m = 0$ (i.e., when there are no clauses with head predicate p) the Clark formula is simply

$$\forall \vec{\mathrm{Y}}. \neg p(\vec{\mathrm{Y}})$$

This can be understood as a genuine special case of the previous definition, since the correct definition of an empty disjuction is the truth value **false**.

General completion procedure: Step 3

The *Clark completion*, $Comp(T)$ of the definite clause theory T is the theory consisting of:

- Clark formulas for every predicate p appearing in the theory T.
- $\neg(t_1 = t_2)$ for every pair t_1, t_2 of non-unifiable terms.

Clark completion of Lecture 7 example

As another illustrative example, here is the Clark completion of the example from Lecture 7.

$$\forall \mathtt{X}.\, \mathtt{nasty(X)} \leftrightarrow \mathtt{cheap(X)}$$

$$\forall \mathtt{X}.\, \mathtt{cool(X)} \leftrightarrow (\mathtt{free(X)} \vee \mathtt{X = mac})$$

$$\forall \mathtt{X}.\, \mathtt{cheap(X)} \leftrightarrow \mathtt{X = windows}$$

$$\forall \mathtt{X}.\, \mathtt{free(X)} \leftrightarrow \mathtt{X = linux}$$

$$\mathtt{linux} \neq \mathtt{mac} \quad \mathtt{mac} \neq \mathtt{windows} \quad \mathtt{windows} \neq \mathtt{linux}$$

Properties of Clark completion

1. The theory $Comp(T)$ extends T.
   That is, $T \models F$ implies $Comp(T) \models F$, for all formulas F.

2. The theory $Comp(T)$ is consistent.
   Indeed, the minimal Herbrand model of T is a model of $Comp(T)$.

3. If $Comp(T) \models A$, where A is an atomic formula, then $T \models A$.
   (That is, the Clark completion adds no new *positive* information.)

4. If the prolog query $\backslash+\,\mathtt{A}$ succeeds, where A is a ground atomic formula. Then $Comp(T) \models \neg A$.
   (That is, negation by failure for ground queries is sound relative to the Clark completion.)

Two issues with the CWA revisited

1. In contrast to the CWA, one can effectively compute $Comp(T)$ from T. Indeed, the description we have given for this theory essentially gives an algorithm for constructing it.
2. The Clark completion more precisely captures the behaviour of Prolog's negation by failure. Consider again the propositional theory `T` consisting of a single axiom

   $$\mathtt{p} \rightarrow \mathtt{p}$$

   As before, the Prolog query $\backslash+\,\mathtt{p}$ goes into a loop. The Clark completion of this theory is the theory

   $$\mathtt{p} \leftrightarrow \mathtt{p}$$

   Thus $Comp^{+}(T) \not\models \neg\mathtt{p}$, which more closely models Prolog's behaviour.

Clark completion — summary

- The Clark completion of T can be effectively computed from T and soundly models negation by failure.
- It is more faithful to Prolog behaviour on negated queries than the CWA.
- Although it has been described in this lecture for definite clause theories only, the Clark completion can be more generally defined for NAF Prolog programs and goals.