Logic Programming

Theory Lecture 5: (In)completeness for Definite Clause Predicate Logic

Alex Simpson

School of Informatics

28th October 2013

Recap (Lecture 3): Definite clause predicate logic

A *definite clause* is a formula of one of the two shapes below

$$B \qquad \text{(a Prolog } \textit{fact } B\,.\text{)}$$

$$A_1 \wedge \cdots \wedge A_k \;\rightarrow B \qquad \text{(a Prolog } \textit{rule } B \;\text{:-}\; A_1, \ldots, A_k.\text{)}$$

where $A_1, \ldots, A_k, B$ are all *atomic formulas*.

A *logic program* is a list $F_1, \ldots, F_n$ of definite clauses

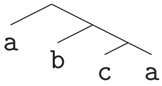
A *goal* is a list $G_1, \ldots, G_m$ of atomic formulas.

The job of the system is to ascertain whether the logical consequence below holds.

$$\forall Vars(F_1).\, F_1, \ldots, \forall Vars(F_n).\, F_n \;\models\; \exists Vars(G_1, \ldots, G_m).\, G_1 \wedge \cdots \wedge G_m$$

Example (Lecture 4): Leaf-labelled binary trees

E.g.



`nd(lf(a),nd(lf(b),nd(lf(c),lf(a))))`

The `path` predicate :

```
path(a,nd(lf(a),nd(lf(b),nd(lf(c),lf(d)))), [l])
path(b, nd(lf(a),nd(lf(b),nd(lf(c),lf(d)))), [r,l])
path(c, nd(lf(a),nd(lf(b),nd(lf(c),lf(d)))), [r,r,l])
path(a, nd(lf(a),nd(lf(b),nd(lf(c),lf(d)))), [r,r,r])
```

Example program, query and result (Lecture 4)

Program

$$
\begin{array}{rcl}
 & & \texttt{path(X,lf(X),[])} \\
\texttt{path(X,T,P)} & \rightarrow & \texttt{path(X,nd(T,U),[l|P])} \\
\texttt{path(X,T,P)} & \rightarrow & \texttt{path(X,nd(S,T),[r|P])}
\end{array}
$$

Goal

`path(b,nd(lf(a),nd(lf(b),lf(c))),Q), path(Y,nd(lf(c),nd(lf(a),lf(b))),Q)`

Result

$$\{\texttt{Q} = \texttt{[r,l]},\ \texttt{Y} = \texttt{a}\}$$

Prolog proof search and inference

Given a program

$$F_1, \ldots, F_n$$

and goal

$$G_1, \ldots, G_m \quad ,$$

a Prolog proof search, if successful, results in a substitution θ, witnessing the implicit existential quantification in the goal.

As in the case of propositional logic (Lectures 1 and 2), it is helpful to understand the proof search procedure as constructing a derivation (proof) in an inference system for definite clause predicate logic.

Inference system (a.k.a. SLD resolution)

As before, we use a single inference rule to derive new *established goals* from goals already established.

$$\frac{(C_1, \ldots, C_{l-1}, A_1, \ldots, A_k, C_{l+1}, \ldots, C_m)\theta \qquad A_1 \wedge \cdots \wedge A_k \;\rightarrow\; B}{C_1, \ldots, C_{l-1},\; C_l,\; C_{l+1}, \ldots, C_m}$$

where:

- $1 \leq l \leq m$,
- $k \geq 0$,
- $A_1 \wedge \cdots \wedge A_k \;\rightarrow\; B$ is one of the program clauses F_i, with its variables renamed to variables not appearing in $C_1, \ldots, C_m$.
- θ is the most general unifier of B and C_l.

A *derivation* (or *proof*) is a sequence of applications of the above rule, in which the topmost rule has the empty goal as its premise.

Technical points

The general format $A_1 \wedge \cdots \wedge A_k \rightarrow B$ of clauses includes Prolog *facts* as the special case in which $k = 0$.

In this case, the inference rule specialises to

$$\frac{(C_1, \ldots, C_{l-1},\ C_{l+1}, \ldots, C_m)\theta \qquad B}{C_1, \ldots, C_{l-1},\ C_l,\ C_{l+1}, \ldots, C_m}$$

Technical Lemma 1
If the two goals $G_1, \ldots, G_m$ and $G'_1, \ldots, G'_n$ are derivable and have no variables in common then the juxtaposed goal $G_1, \ldots, G_m, G'_1, \ldots, G'_n$ is also derivable.

Technical Lemma 2
If a substitution instance $(G_1, \ldots, G_m)\theta$ of a goal $G_1, \ldots, G_m$ is derivable then the goal $G_1, \ldots, G_m$ is itself derivable.

Program:

$$
\begin{array}{rcrcl}
F_1 & = & & & \texttt{path(X,lf(X),[])} \\
F_2 & = & \texttt{path(X,T,P)} & \rightarrow & \texttt{path(X,nd(T,U),[l|P])} \\
F_3 & = & \texttt{path(X,T,P)} & \rightarrow & \texttt{path(X,nd(S,T),[r|P])}
\end{array}
$$

Example derivation:

$$
\begin{array}{cl}
\epsilon & F_1 \\ \hline
\texttt{path}(\texttt{Y},\texttt{lf}(\texttt{a}),[]) & F_2 \\ \hline
\texttt{path}(\texttt{Y},\texttt{nd}(\texttt{lf}(\texttt{a}),\texttt{lf}(\texttt{b})),[\texttt{l}]) & F_3 \\ \hline
\texttt{path}(\texttt{Y},\texttt{nd}(\texttt{lf}(\texttt{c}),\texttt{nd}(\texttt{lf}(\texttt{a}),\texttt{lf}(\texttt{b}))),[\texttt{r},\texttt{l}]) & F_1 \\ \hline
\texttt{path}(\texttt{b},\texttt{lf}(\texttt{b}),\texttt{Q}'')\ ,\ \texttt{path}(\texttt{Y},\texttt{nd}(\texttt{lf}(\texttt{c}),\texttt{nd}(\texttt{lf}(\texttt{a}),\texttt{lf}(\texttt{b}))),[\texttt{r},\texttt{l}|\texttt{Q}'']) & F_2 \\ \hline
\texttt{path}(\texttt{b},\texttt{nd}(\texttt{lf}(\texttt{b}),\texttt{lf}(\texttt{c})),\texttt{Q}')\ ,\ \texttt{path}(\texttt{Y},\texttt{nd}(\texttt{lf}(\texttt{c}),\texttt{nd}(\texttt{lf}(\texttt{a}),\texttt{lf}(\texttt{b}))),[\texttt{r}|\texttt{Q}']) & F_3 \\ \hline
\texttt{path}(\texttt{b},\texttt{nd}(\texttt{lf}(\texttt{a}),\texttt{nd}(\texttt{lf}(\texttt{b}),\texttt{lf}(\texttt{c}))),\texttt{Q})\ ,\ \texttt{path}(\texttt{Y},\texttt{nd}(\texttt{lf}(\texttt{c}),\texttt{nd}(\texttt{lf}(\texttt{a}),\texttt{lf}(\texttt{b}))),\texttt{Q}) &
\end{array}
$$

Prolog proof search finds derivations

Every successful branch in a Prolog search tree gives rise to a derivation in the inference system.

Essentially, we construct the derivation by turning the branch upside down.

This observation underlies:

Proposition
If Prolog proof search, for a program $F_1, \ldots, F_n$ and goal $G_1, \ldots, G_m$ succeeds then the goal has a derivation in the inference system.

Soundness of inference system

Theorem
If goal $G_1, \ldots, G_m$ is derivable in the inference system for a program $F_1, \ldots, F_n$, then

$$\forall Vars(F_1).\, F_1, \ldots, \forall Vars(F_n).\, F_n \models \exists Vars(G_1, \ldots, G_m).\, G_1 \wedge \cdots \wedge G_m$$

Proof
Similar to proof of soundness in the propositional case (Lecture 2), though the definition of logical consequence in terms of structures, rather than truth-value interpretations, adds (significant but tedious) complications.

Soundness of inference system

Theorem

If goal $G_1, \ldots, G_m$ is derivable in the inference system for a program $F_1, \ldots, F_n$, then

$$\forall Vars(F_1).\, F_1, \ldots, \forall Vars(F_n).\, F_n \models \exists Vars(G_1, \ldots, G_m).\, G_1 \wedge \cdots \wedge G_m$$

Proof (Non-examinable)

Similar to proof of soundness in the propositional case (Lecture 2), though the definition of logical consequence in terms of structures, rather than truth-value interpretations, adds (significant but tedious) complications.

Completeness and Incompleteness

Completeness of inference system

For any program $F_1, \ldots, F_n$ and goal $G_1, \ldots, G_m$ such that

$$\forall Vars(F_1).\, F_1, \ldots, \forall Vars(F_n).\, F_n \models \exists Vars(G_1, \ldots, G_m).\, G_1 \wedge \cdots \wedge G_m$$

there exists a derivation of $G_1, \ldots, G_m$ in the inference system ...

Incompleteness of Prolog proof search

... however, Prolog proof search for the goal $G_1, \ldots, G_m$ need not succeed (it may fail to terminate).

Completeness and Incompleteness

Completeness of inference system

For any program $F_1, \ldots, F_n$ and goal $G_1, \ldots, G_m$ such that

$$\forall \mathit{Vars}(F_1).\, F_1, \ldots, \forall \mathit{Vars}(F_n).\, F_n \models \exists \mathit{Vars}(G_1, \ldots, G_m).\, G_1 \wedge \cdots \wedge G_m$$

there exists a derivation of $G_1, \ldots, G_m$ in the inference system ...

Incompleteness of Prolog proof search

... however, Prolog proof search for the goal $G_1, \ldots, G_m$ need not succeed (it may fail to terminate).

The example used to demonstrate incompleteness of propositional Prolog search in Lecture 2 is again an example of the incompleteness of proof search in the case of predicate logic.

The proof of completeness of the inference system is more involved for predicate logic than for propositional logic. We consider this in some detail, since it involves an important construction.

Outline proof of completeness

Suppose that

$$\forall Vars(F_1).\, F_1, \ldots, \forall Vars(F_n).\, F_n \models \exists Vars(G_1, \ldots, G_m).\, G_1 \wedge \cdots \wedge G_m$$

We shall construct a very special *structure* (in the sense of Lecture 3), the *minimal Herbrand model* of the program $F_1, \ldots, F_n$.

Because the minimal Herbrand model $\mathcal{H}$ is a model, we have

$$\mathcal{H} \models \exists Vars(G_1, \ldots, G_m).\, G_1 \wedge \cdots \wedge G_m$$

Due to the cunning way $\mathcal{H}$ is defined, it will follow that the goal $G_1, \ldots, G_m$ is derivable.

Jacques Herbrand (1908–1931)

The Herbrand universe

A term is said to be *ground* if it contains no variables.

$$\begin{aligned} \mathit{ground\_term} \;&::=\; \mathit{constant} \\ &\mid\; \mathit{fn\_symbol}\,(\mathit{ground\_term\_list}) \end{aligned}$$

$$\begin{aligned} \mathit{ground\_term\_list} \;&::=\; \mathit{ground\_term} \\ &\mid\; \mathit{ground\_term}, \mathit{ground\_term\_list} \end{aligned}$$

Note that just the variables have been omitted from the grammar for terms from Lecture 3.

The *Herbrand universe* is just the set of all ground terms.

(We need to ensure that there is at least one constant symbol in our vocabulary in order that the Herbrand universe is non-empty.)

The minimal Herbrand model

We define the structure $\mathcal{H}$ as follows.

- The universe is the Herbrand universe.
- A constant $\mathtt{c}$ is interpreted by $\mathtt{c}^{\mathcal{H}} = \mathtt{c}$.
- A function symbol $\mathtt{f}/k$ is interpreted by $\mathtt{f}^{\mathcal{H}}(u_1, \ldots, u_k) = \mathtt{f}(u_1, \ldots, u_k)$.
- A predicate symbol $\mathtt{p}/k$ is interpreted by

$$\mathtt{p}^{\mathcal{H}}(u_1, \ldots, u_k) = \textbf{true} \iff \text{the goal } \mathtt{p}(u_1, \ldots, u_k) \text{ is derivable}$$

(N.B., we are using $u_1, u_2, \ldots$ to range over ground terms.)

Technical observation

Consider any atomic formula $\mathtt{p}(t_1, \ldots, t_k)$ with $Vars(t_1, \ldots, t_k) = \mathtt{X}_1, \ldots, \mathtt{X}_l$, and let $u_1, \ldots, u_l$ be ground terms.

By definition of $\mathtt{p}^{\mathcal{H}}$, we have that

$$\mathcal{H} \models_{[\mathtt{X}_1 = u_1, \ldots, \mathtt{X}_l = u_l]} \mathtt{p}(t_1, \ldots, t_k)$$

holds if and only if the ground atomic goal below has a derivation.

$$\mathtt{p}(t_1, \ldots, t_k)\{\mathtt{X}_1 = u_1, \ldots, \mathtt{X}_l = u_l\}$$

Proof of completeness (started)

Thus far, we have defined the structure $\mathcal{H}$.

The next step is to show that $\mathcal{H}$ is indeed a model of the program. That is, we show that

$$\mathcal{H} \models \forall Vars(F_i).\, F_i$$

for every F_i in $F_1, \ldots, F_n$.

The detailed argument is given in the Appendix of this lecture.

Now, since

$$\forall Vars(F_1).\, F_1, \ldots, \forall Vars(F_n).\, F_n \models \exists Vars(G_1, \ldots, G_m).\, G_1 \wedge \cdots \wedge G_m$$

and $\mathcal{H} \models \forall Vars(F_1).\, F_i$, for every F_i in $F_1, \ldots, F_n$, it follows that:

$$\mathcal{H} \models \exists Vars(G_1, \ldots, G_m).\, G_1 \wedge \cdots \wedge G_m$$

Proof of completeness (completed)

We have:

$$\mathcal{H} \models \exists Vars(G_1, \ldots, G_m).\, G_1 \wedge \cdots \wedge G_m$$

Let $\mathtt{Y}_1, \ldots, \mathtt{Y}_l$ be $Vars(G_1, \ldots, G_m)$. By definition of satisfaction, there exist ground terms $u_1, \ldots, u_l$ such that

$$\mathcal{H} \models_{[\mathtt{Y}_1 := u_1, \ldots, \mathtt{Y}_l := u_l]} G_1 \wedge \cdots \wedge G_m$$

So, by the technical observation, each of the the atomic formulas $G_i\theta$, where θ is the substitution $\{\mathtt{Y}_1 = u_1, \ldots, \mathtt{Y}_l = u_l\}$, is individually derivable as a ground atomic goal.

By technical lemma 1, it follows that the amalgamated goal

$$G_1\theta, \ldots, G_m\theta\,, \text{ which is the same as } (G_1, \ldots, G_m)\theta$$

is derivable. Whence, by technical lemma 2, so is

$$G_1, \ldots, G_m$$

Complete proof search

We have a complete inference system for programs consisting of definite clauses in predicate logic, but Prolog's proof search procedure is incomplete.

As in propositional logic, incompleteness can be remedied by changing the search strategy.

For example, a complete procedure is obtained by making both the following two changes.

1. Adopt a breadth-first search strategy.
2. Allow the search to choose *any* goal G_i from the current goal list $G_1, \ldots, G_m$ (instead of always choosing G_1)

This provides a complete proof strategy, but *not* a decision procedure. The search does not terminate in cases in which a goal is not provable.

Undecidability

A *decision procedure* for definite clause predicate logic would be an algorithm that, given a goal and a program, outputs **yes** if the goal is a logical consequence of the program, and **no** otherwise. Note that a decision procedure is required to always terminate with one answer or the other.

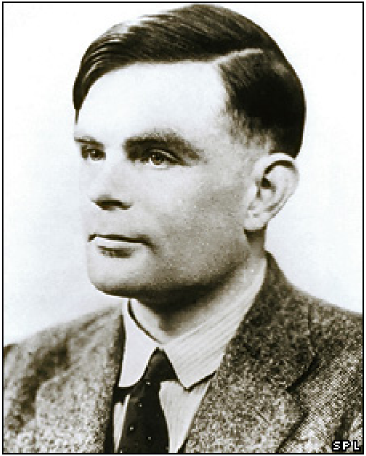
Unlike in propositional logic, it is *impossible* to provide a decision procedure for definite clause predicate logic.

One can show that the *halting problem* (the question of deciding whether the execution of a *Turing machine* eventually halts) can be encoded as a query in definite clause predicate logic.

In fundamental work, in 1936, Alan Turing showed that the halting problem is an *undecidable problem*

(The undecidability of the halting problem is covered in detail in UG3 "Computability and Intractability")

Alan Turing (1912–1954)



Conclusions (reprising Lecture 2)

Prolog proof search is an example of good engineering design based on an interplay between theoretical and practical considerations.

- It has a strong theoretical foundation, in being based on a complete inference system for denite-clause predicate logic.
- However, its incomplete proof search procedure is a good implementation choice, allowing efficient proof search in the context of predicate logic, and permitting the programmer to tailor programs with regard to efficiency issues.

Main points today

inference system (SLD resolution) for definite clause predicate logic

soundness of inference system

completeness of inference system

incompleteness of Prolog proof search strategy

Herbrand universe

minimal Herbrand model

undecidability of definite clause predicate logic

Appendix: The minimal Herbrand model is a model

Suppose F_i is $A_1 \wedge \cdots \wedge A_k \rightarrow B$ with $Vars(F_i) = \mathtt{X}_1, \ldots, \mathtt{X}_l$.
To verify that $\mathcal{H} \models \forall Vars(F_i).\, F_i$, we must show that, for all ground terms $u_1, \ldots, u_l$ it holds that

$$\mathcal{H} \models_{[\mathtt{X}_1 := u_1, \ldots, \mathtt{X}_l := u_l]} A_1 \wedge \cdots \wedge A_k \rightarrow B$$

Suppose then that $\mathcal{H} \models_{[\mathtt{X}_1 := u_1, \ldots, \mathtt{X}_l := u_l]} A_j$ for every $j \in \{1, \ldots, k\}$.
By the technical observation, every ground atomic goal

$$A_j\{\mathtt{X}_1 = u_1, \ldots, \mathtt{X}_l = u_l\}$$

is derivable. Hence, applying technical lemma 1, we can derive the premise whence conclusion of the inference rule below

$$\frac{(A_1, \ldots, A_k)\{\mathtt{X}_1 = u_1, \ldots, \mathtt{X}_l = u_l\} \qquad F_i}{B\{\mathtt{X}_1 = u_1, \ldots, \mathtt{X}_l = u_l\}}$$

Now, by our technical observation, we have as required:

$$\mathcal{H} \models_{[\mathtt{X}_1 := u_1, \ldots, \mathtt{X}_l := u_l]} B$$