INST0072 Lecture 10: Natural Deduction and Further Topics in Knowledge Representation

## 1. In the Last Lecture

- A query execution with a stratified (i.e. non-looping) propositional Prolog program without negation-as-failure corresponds to a search for a resolution proof by contradiction.
- For "sensibly written" (i.e. safe, stratified) Prolog programs with negation-as-failure, their logical meaning can be understood as their Clark completion.
- For predicate Prolog programs, the Clark completion must include Clark equality theory, because of the way Prolog unifies terms and variables.
- In A.I. terms, Clark completion is related to the "closed world assumption", and Clark equality theories are examples of "uniquness-of-names axioms".

## 2. Comparing Resolution and Prolog Proofs

From Slide 8, Lecture 9:

Resolution proof:

(5) $(\neg big)$

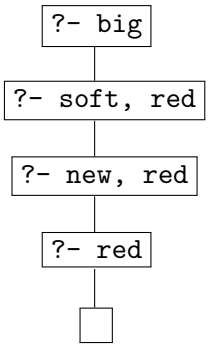
(6) $(\neg soft \vee \neg red)$

(7) $(\neg new \vee \neg red)$

(8) $(\neg red)$

(9) $\bot$

Prolog search tree:



## 3. Example Completion with Equality Theory

From Slide 19, Lecture 9:
Prolog program $PR$:

```
likes(X, partner_of(X)).
likes(nina, X) :- polite(X).
polite(partner_of(mani)).
```

**COMP**($PR$):

$$\forall x_1 \forall x_2.[likes(x_1, x_2) \leftrightarrow [\exists x.(x_1 = x \wedge x_2 = partner\_of(x)) \vee \exists x.(x_1 = nina \wedge x_2 = x \wedge polite(x))]],$$
$$\forall x_1.[polite(x_1) \leftrightarrow x_1 = partner\_of(mani)],$$

Clark equality theory for $PR$:
$$mani \neq nina \wedge \forall x.[partner\_of(x) \neq mani \wedge partner\_of(x) \neq nina],$$
$$\forall x_1 \forall x_2.[x_1 \neq x_2 \rightarrow partner\_of(x_1) \neq partner\_of(x_2)],$$
plus for any structured term $\tau[x]$ containing any variable $x$:
$\tau[x] \neq x$ [e.g. $partner\_of(partner\_of(x)) \neq x$]

## 4. Calculi - Some Reminders

- An *inference rule* $\frac{P_1 \ldots P_n}{C}$ generates a conclusion $C$ from a collection of premises $P_1 \ldots P_n$.
- A set of inference rules is called a *calculus*, and a sequence of applications of inference rules is called a *derivation*.
- "$\vdash$" means "can derive". "$KB \vdash F$" means "there is a derivation of $F$ from the knowledge base or theory $KB$".
- "$\models$" means "entails". "$KB \models F$" means "every model of $KB$ is a model of $F$".
- A calculus is *sound* if whenever $KB \vdash F$ then $KB \models F$.
- A calculus is *complete* if whenever $KB \models F$ then $KB \vdash F$.

## 5. Calculi for Predicate Logic

- Two commonly used calculi for predicate logic are *resolution* and *natural deduction*.
- We have already covered resolution for propositional logic in Lectures 1 and 3. This can be extended to predicate logic using a notion of *unification of terms* similar in some ways to unification in Prolog (see e.g. Ertel Chapter 3, Section 3.5).
- Historically, resolution has been used as a basis for automated deduction.
- In contrast, natural deduction is traditionally regarded as "natural" for humans to use and understand, since its relatively large collection of inference rules corresponds to an intuitive understanding of logical quantifiers and connectives.

## 6. Natural Deduction

- Full technical coverage of natural deduction is beyond the scope of this module – we will just look at a few example inference rules and derivations.
- In natural deduction, each logical connective and quantifier has *introduction rules* and *elimination rules*.
- Many versions of natural deduction also include some additional rules to deal with equality.
- Some versions of natural deduction allow the use of *lemmas* (previously derived tautologies) such as $F \vee \neg F$ in derivations.
- Derivations can include *sub-derivations*, often separated out in a "box", to allow extra temporary assumptions. This allows the calculus to model human reasoning strategies such as "proof by cases" and "proof by contradiction".
- Natural deduction is both sound and complete.

## 7. Natural Deduction Rules For $\wedge$

- The $\wedge$ connective has one introduction rule:

$$\frac{A, \quad B}{A \wedge B} \qquad \wedge\text{-introduction (or '}\wedge I\text{' for short)}$$

- and two elimination rules:

$$\frac{A \wedge B}{A} \qquad \frac{A \wedge B}{B} \qquad \wedge\text{-elimination ('}\wedge E\text{')}$$

- An example derivation showing $\{P, \neg Q, R\} \vdash (P \wedge \neg Q) \wedge R$:

| | | |
|---|---|---|
| 1 | $P$ | assumption |
| 2 | $\neg Q$ | assumption |
| 3 | $R$ | assumption |
| 4 | $P \wedge \neg Q$ | $\wedge I(1, 2)$ |
| 5 | $(P \wedge \neg Q) \wedge R$ | $\wedge I(4, 3)$ |

## 8. Natural Deduction Rules For $\rightarrow$

- The $\rightarrow$ connective has one elimination rule (otherwise known as *Modus Ponens*):

$$\frac{A, \quad A \rightarrow B}{B} \qquad \rightarrow\text{-elimination ('}\rightarrow E\text{')}$$

- and one introduction rule:

$$\frac{\begin{array}{c}[\text{A}] \\ \vdots \\ \text{B}\end{array}}{A \rightarrow B} \qquad \rightarrow\text{-introduction ('}\rightarrow I\text{')}$$

- The top half of the $\rightarrow I$ rule means "temporarily assuming $A$ allows a sub-derivation of $B$". The sub-derivation steps cannot be used in other parts of the derivation, because they rely on an extra temporary assumption. So they are put in a "box". Immediately after the box the assumption $A$ is "discharged".

## 9. An Example Derivation Using $\rightarrow E$ and $\rightarrow I$

$$\frac{A, \quad A \rightarrow B}{B} \quad (\rightarrow E) \qquad\qquad \frac{\begin{array}{c}[A] \\ \vdots \\ B\end{array}}{A \rightarrow B} \quad (\rightarrow I)$$

An example showing that $\{P \rightarrow \neg Q, \neg Q \rightarrow R\} \vdash P \rightarrow R$:

| | | |
|---|---|---|
| 1 | $P \rightarrow \neg Q$ | assumption |
| 2 | $\neg Q \rightarrow R$ | assumption |
| 3 | $P$ | assumption |
| 4 | $\neg Q$ | $\rightarrow E(1,3)$ |
| 5 | $R$ | $\rightarrow E(2,4)$ |
| 6 | $P \rightarrow R$ | $\rightarrow I(3,5)$ |

Note that steps 3, 4 and 5 can ony be used for step 6, after which step 3 is discharged.

## 10. Natural Deduction Rules For $\forall$

In the $\forall$-elimination and $\forall$-introduction rules below:

- $A_x$ is any formula containing free (i.e. unquantified) occurrences of the variable $x$,
- $c$ is an extra new constant symbol temporarily introduced into the logic's signature for a (boxed) sub-proof,
- $g$ is any ground term of the signature (i.e. any term not containing any variables),
- $A_x(c/x)$ is identical to the formula $A_x$ but with all free occurrences of $x$ replaced by the constant $c$, and
- $A_x(g/x)$ is identical to the formula $A_x$ but with all free occurrences of $x$ replaced by the ground term $g$.

$$\frac{\forall x.A_x}{A_x(g/x)} \quad (\forall E) \qquad\qquad \frac{\begin{array}{c}[c] \\ \vdots \\ A_x(c/x)\end{array}}{\forall x.A_x} \quad (\forall I)$$

## 11. An Example Derivation Using $\forall E$ and $\forall I$

$$\frac{\forall x.A_x}{A_x(g/x)}\ (\forall E) \qquad\qquad \frac{\begin{array}{c}[c]\\ \vdots \\ A_x(c/x)\end{array}}{\forall x.A_x}\ (\forall I)$$

An example showing $\{\forall x.(P(x) \rightarrow Q(x)), \forall x.P(x)\} \vdash \forall x.Q(x)$:

| | | |
|---|---|---|
| 1 | $\forall x.(P(x) \rightarrow Q(x))$ | assumption |
| 2 | $\forall x.P(x)$ | assumption |
| 3 | $c$ | $\forall I$ constant |
| 4 | $P(c)$ | $\forall E(2,3)$ |
| 5 | $P(c) \rightarrow Q(c)$ | $\forall E(1,3)$ |
| 6 | $Q(c)$ | $\rightarrow E(4,5)$ |
| 7 | $\forall x.Q(x)$ | $\forall I(3,6)$ |

As before, steps 3 – 6 can ony be used for step 7.

## 12. Natural Deduction – Where to Find Out More

It is quite difficult to find a detailed, straightforward and clear description of natural deduction, and there are many variations on the way derivations are written out. But if you are interested in finding out more, the following sources might help:

- Part II of "Reasoned Programming", by Broda, Eisenbach, Khoshnevisan & Vickers, pub. Prentice Hall, ISBN 0-13-098831-6. (This book is expensive, so it's better to borrow it from the library.)
- James Studd's lectures at: http://users.ox.ac.uk/~logicman/. His "slides week 6" are on natural deduction, but using a different (equivalent) way to display derivations. You might find his other lectures useful as well.

## 13. Logic and Knowledge Representation – Topics We Would Cover Next Given More Time

- Many-sorted and modal logics.
- Default reasoning and non-monotonic logics.
- Anwser Set Programming.
- Argumentation and Argumentation Logic.

## 14. Many-sorted and Modal Logics

- *Many-sorted logic* is an extension of predicate logic that partitions the domain of discourse into different sub-sets, each of which is a particular "sort" or "type". We can then specify the sort of each constant symbol and each argument of each function and predicate.
- For example, if we have a predicate symbol *eats*/2, we can specify in the logic's signature that the first argument is of sort "person" and the second argument is of sort "food".
- In *modal logic* a model is a connected structure of *possible worlds*, only one of which is the "actual world".
- Given a formula $F$, the modal formula $\Box F$ means "$F$ is necessarily true" (true in all possible worlds connected to the actual world), and the modal formula $\Diamond F$ means "$F$ is possibly true" (true in at least one connected world).

## 15. Default Reasoning and Non-monotonic Logics

- Humans often assume extra information from things they are told. For example, if told "Tweety is a bird", people assume that Tweety can fly, because most (but not all) birds can fly.
- Default reasoning is concerned with developing logics and logical systems that can express and reason with rules such as "most birds fly".
- Two well-known logics that can model default reasoning are Ray Reiter's *Default Logic* and John MacCarthy's idea of *Circumscription*.
- Default reasoning systems are often described as "non-monotonic" because adding new facts can stop us from deriving previous conclusions. If we are also told "Tweety is a penguin" we can no longer conclude "Tweety can fly".

## 16. Anwser Set Programming

- Answer set programming (ASP) is a form of logic programming particularly useful to solve problems involving large search spaces.
- ASP programs look like Prolog programs, but the underlying computation mechanism is very different, and based on generating alternative models, using a semantics called *stable model theory*.
- ASP programs can include both negation-as-failure and classical negation.
- For further explanation see e.g. Vladimir Lifshitz's article "What Is Answer Set Programming?". For a working ASP system, visit `https://potassco.org`.

## 17. Argumentation and Argumentation Logic

- *Argumentation* is a very broad area of study encompasing, for example, theories of logic, dialogue, conversation, and persuasion.
- A.I. has used ideas from argumentation theory to develop knowledge representation systems, including, for example, *argumentation logic (AL)*.
- In AL, *arguments* are regarded as sets of propositions of different *strengths*, and different arguments may *attack* each other. An argument is valid if it is able to *defend* itself against attacks.
- For further information on AL and its relationship to classical logic see for example "On Argumentation Logic and Propositional Logic" by Kakas et al.