Theory 3: Practical Week 9

Proofs in predicate logic

Predicate logic extends propositional logic with the universal and existential quantifiers. Instead of propositional variables, we are now dealing with predicates. All the rules from propositional logic still apply, but there are now introduction and elimination rules for quantifiers and equality.

- Predicate symbols are written as capital letters
- The universal quantifier ∀ can be written as `A` or `@`, and the existential ∃ as `E` or `3`.
- This is the same for rule names, so that ∀E is `AE` and ∃I is `EI`.
- When binding a variable with a quantifier, Carnap only allows letters in the range `s-z` to be used
- When using a fixed constant that is not bound by a quantifier, Carnap only allows letters in the range `a-r`

Then, we can write the sentence $\forall x. \exists y. Q(y) \rightarrow P(x,a)$ as `AxEy(Q(y)->P(x,a))`.

Exercise 1 - Universal Elimination

The universal elimination rule lets us specialise a universally quantified formula to use a given value. Here's an example usage:

Playground

```
1. a = b :PR
2. b = b :=I
   b = a :=E 1 2
```

Prove the following lemmas using the `AE` rule and rules from propositional logic.

1.2

∀xP(x) ⊢ P(a) ∧ P(b) ✓

```
1. Ax P(x) :PR
2. P(a) :AE 1
3. P(b) :AE 1
4. P(a) & P(b) :&I 2, 3
```

1.	∀xP(x)	PR
2.	P(a)	∀E 1
3.	P(b)	∀E 1
4.	P(a) ∧ P(b)	∧I 2, 3

1.3

∀x(P(x) → Q(x)), ¬Q(a) ⊢ ¬P(a) ✓

```
1. Ax(P(x) → Q(x)) :PR
2. ~Q(a) :PR
3. P(a) → Q(a) :AE 1
4. ~P(a) :MT 3, 2
```

1.	∀x(P(x) → Q(x))	PR
2.	¬Q(a)	PR
3.	P(a) → Q(a)	∀E 1
4.	¬P(a)	MT 3, 2

1.4

∀x((P(x) ∧ Q(x)) → ∀zR(x,z)), P(a), Q(a) ⊢ R(a,b) ✓

```
1. Ax((P(x) & Q(x)) → AzR(x, z)) :PR                    +
2. P(a) :PR                                              +
3. Q(a) :PR                                              +
4. P(a) & Q(a) :&I 2, 3                                  +
5. (P(a) & Q(a)) → AzR(a,z) :AE 1                        +
6. AzR(a, z) :→E 5, 4                                    +
7. R(a, b) :AE 6                                         +
```

1.	∀x((P(x) ∧ Q(x)) → ∀zR(x,z))	PR
2.	P(a)	PR
3.	Q(a)	PR
4.	P(a) ∧ Q(a)	∧I 2, 3
5.	(P(a) ∧ Q(a)) → ∀zR(a,z)	∀E 1
6.	∀zR(a,z)	→E 5, 4
7.	R(a,b)	∀E 6

1.5

∀x(M(x) → R(x)), M(a) ⊢ R(a) ✓

```
1. Ax(M(x) → R(x)) :PR                                   +
2. M(a) :PR                                              +
3. M(a) → R(a) :AE 1                                     +
4. R(a) :→E 3, 2                                         +
```

1.	∀x(M(x) → R(x))	PR
2.	M(a)	PR
3.	M(a) → R(a)	∀E 1
4.	R(a)	→E 3, 2

1.6

```
∀xL(x,b), ∀y(L(b,y) → y=m) ⊢ b=m                                   ✓

1. AxL(x, b) :PR                                                   +
2. Ay(L(b, y) → y=m) :PR                                           +
3. L(b, b) :AE 1                                                   +
4. L(b, b) → b=m :AE 2                                             +
5. b=m :→E 4, 3                                                    +
```

1.	∀xL(x,b)	PR
2.	∀y(L(b,y) → y=m)	PR
3.	L(b,b)	∀E 1
4.	L(b,b) → b=m	∀E 2
5.	b=m	→E 4, 3

Exercise 2 - Existential Introduction

To prove an existentially quantified statement, we need to give a witness. The rule for existential introduction states that if we can show that some property `P` holds for some value `a`, then $\exists x.P(x)$ follows.

Here is an example:

Playground

```
P(a) ∧ P(b) ⊢ ∃xP(x)

1. P(a) ∧ P(b) :PR                                                 +
2. P(a) :∧E 1                                                      +
3. ExP(x) :EI 2                                                    +
```

1.	P(a) ∧ P(b)	PR
2.	P(a)	∧E 1
3.	∃xP(x)	∃I 2

Prove the following lemmas using the `EI` rule:

2.1

∀xP(x) ⊢ ∃xP(x)	✓

```
1. AxP(x) :PR              +
2. P(a) :AE 1              +
3. ExP(x) :EI 2            +
```

1.	∀xP(x)	PR
2.	P(a)	∀E 1
3.	∃xP(x)	∃I 2

2.2

∀x(P(x) → Q(x)), P(a) ⊢ ∃xQ(x)	✓

```
1. Ax(P(x) → Q(x)) :PR     +
2. P(a) :PR                +
3. P(a) → Q(a) :AE 1       +
4. Q(a) :→E 3, 2           +
5. ExQ(x) :EI 4            +
```

1.	∀x(P(x) → Q(x))	PR
2.	P(a)	PR
3.	P(a) → Q(a)	∀E 1
4.	Q(a)	→ E 3, 2
5.	∃xQ(x)	∃I 4

2.3

`P(a) ∨ P(b) ⊢ ∃xP(x)` ✓

```
1. P(a) | P(b) :PR
2.     P(a) :AS
3.     ExP(x) :EI 2
4. --
5.     P(b) :AS
6.     ExP(x) :EI 5
7. ExP(x) :|E 1, 2-3, 5-6
```

1.	P(a) ∨ P(b)	PR
2.	P(a)	AS
3.	∃xP(x)	∃I 2
5.	P(b)	AS
6.	∃xP(x)	∃I 5
7.	∃xP(x)	∨E 1, 2-3, 5-6

2.4

`¬P(a) ⊢ ∃x(P(x) → Q(x))` ✓

```
1. ~P(a) :PR
2.     P(a) :AS
3.     !? :~E 1, 2
4.     Q(a) :X 3
5. P(a) -> Q(a) :->I 2-4
6. Ex(P(x) -> Q(x)) :EI 5
```

1.	¬P(a)	PR
2.	P(a)	AS
3.	⊥	¬E 1, 2
4.	Q(a)	⊥E 3
5.	P(a) → Q(a)	→I 2-4
6.	∃x(P(x) → Q(x))	∃I 5

Exercise 3 - Universal Introduction

Universal introduction in Carnap has a slightly different form to the one in our lectures, and in *Logic in Computer Science*. There is no need to introduce a subproof, and instead we simply need to use a fresh constant, for example `a`, which stands for our hypothetical individual. We then use it to prove a formula `ϕ` in terms of `a`. Provided this constant is not used elsewhere, we can then prove `∀x ϕ[a↦x]`. This is equivalent to what we do in the lectures, just without the subproof box. An example is shown below:

Playground

```
1. AxP(x) :PR
2. AxQ(x) :PR
3. P(a) :AE 1
4. Q(a) :AE 2
5. P(a) ∧ Q(a) :∧I 3,4
6. Ax(P(x) ∧ Q(x)) :AI 5
```

Here, we introduce the hypothetical individual `a`, which is not used anywhere else, and use it to specialise the two universal statements. We then combine these in a conjunctive formula, and finally generalise the property using `AI`. The parameter of `AI` is just the line where the final property involving the hypothetical is proved, rather than a range of subproof lines.

Now, prove the following lemmas using rule `AI`. The symbol `T` is used to denote an empty set of premises -- i.e. an argument with a conclusion, but with no premises.

3.1

∀x∀yP(x,y) ⊢ ∀y∀xP(x,y) ✓

```
1. AxAyP(x, y) :PR
2. AyP(a, y) :AE 1
3. P(a, b) :AE 2
4. AxP(x, b) :AI 3
5. AyAxP(x, y) :AI 4
```

1.	∀x∀yP(x,y)	PR
2.	∀yP(a,y)	∀E 1
3.	P(a,b)	∀E 2
4.	∀xP(x,b)	∀I 3
5.	∀y∀xP(x,y)	∀I 4

3.2

```
T ⊢ ∀x(P(x) ∨ ¬P(x))                                            ✓

1.     P(a) :AS                                                 +
2.     P(a) | ~P(a) :|I 1                                       +
3. --
4.     ~P(a) :AS                                                +
5.     P(a) | ~P(a) :|I 4                                       +
6. P(a) | ~P(a) :LEM 1-2, 4-5                                   +
7. Ax(P(x) | ~P(x)) :AI 6                                       +
```

1.	│ P(a)	AS
2.	│ P(a) ∨ ¬P(a)	∨I 1
4.	│ ¬P(a)	AS
5.	│ P(a) ∨ ¬P(a)	∨I 4
6.	P(a) ∨ ¬P(a)	LEM 1-2, 4-5
7.	∀x(P(x) ∨ ¬P(x))	∀I 6

3.3

```
∀x(P(x) → Q(x)), ∀x(Q(x) → R(x)) ⊢ ∀x(P(x) → R(x))              ✓

1. Ax(P(x) → Q(x)) :PR                                          +
2. Ax(Q(x) → R(x)) :PR                                          +
3. P(a) → Q(a) :AE 1                                            +
4. Q(a) → R(a) :AE 2                                            +
5.     P(a) :AS                                                 +
6.     Q(a) :→E 3, 5                                            +
7.     R(a) :→E 4, 6                                            +
8. P(a) → R(a) :→I 5-7                                          +
9. Ax(P(x) → R(x)) :AI 8                                        +
```

1.	∀x(P(x) → Q(x))	PR
2.	∀x(Q(x) → R(x))	PR
3.	P(a) → Q(a)	∀E 1
4.	Q(a) → R(a)	∀E 2
5.	│ P(a)	AS
6.	│ Q(a)	→E 3, 5
7.	│ R(a)	→E 4, 6
8.	P(a) → R(a)	→I 5-7
9.	∀x(P(x) → R(x))	∀I 8

3.4

∀x(P(x) ∨ Q(x)), ∀x(P(x) → R(x)), ∀x(Q(x) → R(x)) ⊢ ∀xR(x) ✓

```
 1. Ax(P(x) | Q(x)) :PR                                     +
 2. Ax(P(x) -> R(x)) :PR                                    +
 3. Ax(Q(x) -> R(x)) :PR                                    +
 4. P(a) | Q(a) :AE 1                                       +
 5. P(a) -> R(a) :AE 2                                      +
 6. Q(a) -> R(a) :AE 3                                      +
 7.     P(a) :AS                                            +
 8.     R(a) :->E 5, 7                                      +
 9. --
10.     Q(a) :AS                                            +
11.     R(a) :->E 6, 10                                     +
12. R(a) :|E 4, 7-8, 10-11                                  +
13. AxR(x) :AI 12                                           +
```

1.	∀x(P(x) ∨ Q(x))	PR
2.	∀x(P(x) → R(x))	PR
3.	∀x(Q(x) → R(x))	PR
4.	P(a) ∨ Q(a)	∀E 1
5.	P(a) → R(a)	∀E 2
6.	Q(a) → R(a)	∀E 3
7.	P(a)	AS
8.	R(a)	→E 5, 7
10.	Q(a)	AS
11.	R(a)	→E 6, 10
12.	R(a)	∨E 4, 7-8, 10-11
13.	∀xR(x)	∀I 12

Exercise 4 - Existential Elimination

An existentially quantified statement of the form $\exists x.P(x)$ tells us there is some constant for which $P(x)$ holds. The existential elimination rule lets us give this constant a name.

In order to do this, we need to start a new subproof which limits the scope of this variable. The existential eliminiation rule takes two parameters - the existentially quantified statement, and a subproof where we give the existentially quantified constant a name, and prove some property using it.

Here's an example:

Playground

```
1. ExP(x) :PR
2. Ax(P(x)→Q(x)) :PR
3.  P(a) :AS
4.  P(a) → Q(a) :AE 2
5.  Q(a) :→E 3,4
6.  ExQ(x) :EI 5
7. ExQ(x) :EE 1,3-6
```

Prove the following lemmas using rule `EE`:

4.1

∃xP(x), ∀x∀y(P(x) → Q(y)) ⊢ ∀yQ(y) ✓

```
1. ExP(x) :PR                     +
2. AxAy(P(x) → Q(y)) :PR          +
3.     P(a) :AS                   +
4.     Ay(P(a) → Q(y)) :AE 2      +
5.     P(a) → Q(b) :AE 4          +
6.     Q(b) :→E 3, 5              +
7.     AyQ(y) :AI 6               +
8. AyQ(y) :EE 1, 3-7              +
```

1.	$\exists x P(x)$	PR
2.	$\forall x \forall y (P(x) \rightarrow Q(y))$	PR
3.	│ $P(a)$	AS
4.	│ $\forall y (P(a) \rightarrow Q(y))$	∀E 2
5.	│ $P(a) \rightarrow Q(b)$	∀E 4
6.	│ $Q(b)$	→E 3, 5
7.	│ $\forall y Q(y)$	∀I 6
8.	$\forall y Q(y)$	∃E 1, 3-7

4.2

∃x(P(x) ∨ Q(x)), ∀x(P(x) → R(x)), ∀x(Q(x) → R(x)) ⊢ ∃xR(x) ✓

```
 1. Ex(P(x) | Q(x)) :PR                                  +
 2. Ax(P(x) → R(x)) :PR                                  +
 3. Ax(Q(x) → R(x)) :PR                                  +
 4.     P(a) | Q(a) :AS                                  +
 5.         P(a) :AS                                     +
 6.         P(a) → R(a) :AE 2                            +
 7.         R(a) :→E 5 6                                 +
 8.     --
 9.         Q(a) :AS                                     +
10.         Q(a) → R(a) :AE 3                            +
11.         R(a) :→E 9 10                                +
12.     R(a) :|E 4 5-7 9-11                              +
13.     ExR(x) :EI 12                                    +
14. ExR(x) :EE 1 4-13                                    +
```

1.	∃x(P(x) ∨ Q(x))	PR
2.	∀x(P(x) → R(x))	PR
3.	∀x(Q(x) → R(x))	PR
4.	P(a) ∨ Q(a)	AS
5.	P(a)	AS
6.	P(a) → R(a)	∀E 2
7.	R(a)	→E 5, 6
9.	Q(a)	AS
10.	Q(a) → R(a)	∀E 3
11.	R(a)	→E 9, 10
12.	R(a)	∨E 4, 5-7, 9-11
13.	∃xR(x)	∃I 12
14.	∃xR(x)	∃E 1, 4-13

4.3

∃x∀yR(x,y) ⊢ ∃xR(x,x) ✓

```
1. ExAyR(x, y) :PR                  +
2.     AyR(a, y) :AS                +
3.     R(a, a) :AE 2                +
4.     ExR(x, x) :EI 3              +
5. ExR(x, x) :EE 1 2-4              +
```

1.	∃x∀yR(x,y)	PR
2.		∀yR(a,y)	AS
3.		R(a,a)	∀E 2
4.		∃xR(x,x)	∃I 3
5.	∃xR(x,x)	∃E 1, 2-4

4.4

∃x(S → Q(x)) ⊢ S → ∃xQ(x) ✓

```
1. Ex(S → Q(x)) :PR                 +
2.     S → Q(a) :AS                 +
3.         S :AS                    +
4.         Q(a) :→E 2 3             +
5.         ExQ(x) :EI 4             +
6.     S → ExQ(x) :→I 3-5           +
7. S → ExQ(x) :EE 1 2-6             +
```

1.	∃x(S → Q(x))	PR
2.		S → Q(a)	AS
3.			S	AS
4.			Q(a)	→E 2, 3
5.			∃xQ(x)	∃I 4
6.		S → ∃xQ(x)	→I 3-5
7.	S → ∃xQ(x)	∃E 1, 2-6

Exercise 5 - Equality

As with all other operators, we have two rules for equality. `=I` lets us construct terms of the form $a = a$, while `=E` lets us apply a substitution $\phi[x{\mapsto}t_2]$ if we know that $t_1 = t_2$ and $\phi[x{\mapsto}t_1]$ holds.

Here's an example:

Playground

```
1. a = b :PR
2. b = b :=I
3. b = a :=E 1 2
```

Prove the following lemmas using `=I` and `=E`:

5.1

⊤ ⊢ ∀x∃yx=y ✓

```
1. a=a :=I
2. Eya=y :EI 1
3. AxEyx=y :AI 2
```

1.	a=a	=I
2.	∃ya=y	∃I 1
3.	∀x∃yx=y	∀I 2

5.2

a=b, b=c ⊢ a=c ✓

```
1. a=b :PR
2. b=c :PR
3. a=c :=E 2 1
```

1.	a=b	PR
2.	b=c	PR
3.	a=c	=E 2, 1

5.3

∀x∀y(P(x,y) ↔ x=y) ⊢ P(a,a) ✓

```
1. AxAy(P(x, y) ↔ x=y) :PR
2. Ay(P(a, y) ↔ a=y) :AE 1
3. P(a, a) ↔ a=a :AE 2
4. a=a :=I
5. P(a, a) :↔E 3 4
```

1.	∀x∀y(P(x,y) ↔ x=y)	PR
2.	∀y(P(a,y) ↔ a=y)	∀E 1
3.	P(a,a) ↔ a=a	∀E 2
4.	a=a	=I
5.	P(a,a)	↔E 3, 4

Exercise 6*: De Morgan laws for quantifiers

We already considered the De Morgan laws for conjunction and disjunction. Recall from the lectures that universal quantification is like an infinite conjunction, and existential quantification is like infinite disjunction. Knowing this, we can give analogues of the De Morgan laws, which you should go ahead and prove below. We have restricted the logic so that derived laws like `MT` and `LEM` are not available, so you will need to restrict yourself to the core laws.

6.1

¬∃xP(x) ⊢ ∀x¬P(x) ✓

```
1. ~ExP(x) :PR              +
2.     ~Ax~P(x) :AS         +
3.         P(a) :AS         +
4.         ExP(x) :EI 3     +
5.         !? :~E 1 4       +
6.     ~P(a) :~I 3-5        +
7.     Ax~P(x) :AI 6        +
8.     !? :~E 2 7           +
9. Ax~P(x) :PBC 2-8        +
```

1.	¬∃xP(x)	PR
2.	¬∀x¬P(x)	AS
3.	P(a)	AS
4.	∃xP(x)	∃I 3
5.	⊥	¬E 1, 4
6.	¬P(a)	¬I 3-5
7.	∀x¬P(x)	∀I 6
8.	⊥	¬E 2, 7
9.	∀x¬P(x)	IP 2-8

6.2

¬∀xP(x) ⊢ ∃x¬P(x) ✓

```
1. ~AxP(x) :PR                    +
2.     ~Ex~P(x) :AS               +
3.         ~P(a) :AS              +
4.         Ex~P(x) :EI 3          +
5.         !? :~E 2 4             +
6.     P(a) :PBC 3-5              +
7.     AxP(x) :AI 6               +
8.     !? :~E 1 7                 +
9. Ex~P(x) :PBC 2-8              +
```

1.	¬∀xP(x)	PR
2.	¬∃x¬P(x)	AS
3.	¬P(a)	AS
4.	∃x¬P(x)	∃I 3
5.	⊥	¬E 2, 4
6.	P(a)	IP 3-5
7.	∀xP(x)	∀I 6
8.	⊥	¬E 1, 7
9.	∃x¬P(x)	IP 2-8

This work has been supported by a Munson-Simu Faculty Star award at Kansas State University.