CHAPTER 10 – PREDICATE LOGIC: DERIVATIONS

SECTION 10.1 THE DERIVATION SYSTEM *PD*

- *PD* (predicate derivations) is a natural deduction system which, like SD, has two rules for each logical operator. *PD* is complete and sound, i.e. for any set Γ of sentences of *PL* and any sentence **P** of *PL*, $\Gamma \vDash$ **P** if and only if $\Gamma \vdash$ **P** in *PD*.

- All the derivation rules of *SD* apply to sentences of *PD*. Hence the strategies used in *SD* will also be useful in *PD*.

- *PD* has four new rules, and we'll have some additional strategies too.
- Our new rules are: Universal Elimination, Universal Introduction, Existential Elimination, and Existential Introduction. These rules are intended to account for quantified sentences.

SECTION 10.1 THE DERIVATION SYSTEM *PD*

- *Universal Elimination* (∀E)

  (∀x)**P**

  P(**a**/**x**)

- \` **P**(**a**/**x**)’ stands for the substitution instance of the quantified sentence (∀x)**P.** We get it by dropping the initial quantifier and replacing every instance of **x** with **a** (where \`**a**’ is the **instantiating constant**).
- Universal elimination says that we’re allowed to infer from a universally quantified sentence to any substitution instance of that sentence.

SECTION 10.1 THE DERIVATION SYSTEM *PD*

- *E.g.* All philosophers are somewhat strange. Socrates is a philosopher. Therefore, Socrates is somewhat strange.
- We can symbolize the first claim with `$(\forall x)(Px \supset Sx)$' and the second with `Ps'. Here is a derivation then from the premises to the conclusion.

1	$(\forall x)(Px \supset Sx)$	Ass.
2	Ps	Ass.
3	$Ps \supset Ss$	1 $\forall$E
4	Ss	2, 3 $\supset$E

SECTION 10.1 THE DERIVATION SYSTEM *PD*

- New strategy: When using Universal Elimination use goal sentences as guides to which constant to use in forming the substitution instance of the universally quantified sentence.
- E.g. in the above derivation, we could have used \`a' instead of \`s', but then we wouldn't haven't gotten our goal, which was \`Ss'.
- It doesn't matter if the instantiating constant has already occurs in the quantified sentence. E.g. both of the following are okay:

1	(∀x) (Lxa)	Ass.
2	Lta	1 ∀E

1	(∀x) (Lxa)	Ass.
2	Laa	1 ∀E

SECTION 10.1 THE DERIVATION SYSTEM *PD*

- *Existential Introduction* (∃I)

  P(**a**/**x**)

  (∃x) **P**

1	Fa	Ass.
2	(∃y) Fy	∃I

If the thing designated by the constant `a’ is F, then at least one thing is F.

SECTION 10.1 THE DERIVATION SYSTEM *PD*

- *E.g.* If Alfred is a father, then it follows that someone is a father.

1.	Fa	Ass.
2.	(∃y) Fy	1 ∃I

1	Faa	Ass.
2	(∃y) Fya	1 ∃I
3	(∃y) Fyy	1 ∃I
4	(∃y) Fay	1 ∃I

When the goal to be derived is an existentially quantified sentence establish a substitution instance of that sentence as a subgoal, with the intent of applying Existential Introduction to that subgoal to obtain the goal.

SECTION 10.1 THE DERIVATION SYSTEM *PD*

- *Universal Introduction* (∀I)

  P(**a**/**x**)

  (∀x)**P**

provided that:

I. **a** does not occur in an open assumption

II. **a** does not occur in (∀x)**P**

E.g.: Derive (∀x) Fx

1	(∀y) Fy	Ass.
2	Fb	1 ∀E
3	(∀x) Fx	2 ∀I

SECTION 10.1 THE DERIVATION SYSTEM *PD*

E.g.: Derive $(\forall x)\ [Fx \supset (Fx \vee Gx)]$

1	Fc	A / $\supset$I
2	$Fc \vee Gc$	1 $\vee$I
3	$Fc \supset (Fc \vee Gc)$	1-2 $\supset$I
4	$(\forall x)\ [Fx \supset (Fx \vee Gx)]$	3 $\forall$I

E.g.: mistake in deriving $(\forall y)\ (Fy)$:

1	$Fb\ \&\ \sim Fc$	Ass.
2	Fb	1 &E
3	$(\forall y)\ Fy$	2 $\forall$I WRONG!

SECTION 10.1 THE DERIVATION SYSTEM *PD*

- The previous example violated the first restriction. Here's an example of a derivation that violates the second restriction.

E.g.: mistake in deriving (∀x) (Lxh) :

1	(∀x) Lxx	Ass.
2	Lhh	1 ∀E
3	(∀x) Lxh	2 ∀I WRONG!

The instantiating constant `h' in the substitution instance on line 2 occurs in the sentence we tried to derive by UI on line 3, which violate the second restriction.

SECTION 10.1 THE DERIVATION SYSTEM *PD*

- New strategy when using Universal Introduction: When the current goal is a universally quantified sentence, make a substitution instance of that quantified sentence a subgoal, with the intent of applying UI to derive the goal from the subgoal.
- Make sure that the two restrictions on UI will be met: use an instantiating constant in the substitution instance that does not occur in the universally quantified goal sentence and that does not occur in any assumption that is open at the line where the substitution instance is entered.

- *Existential Elimination* (∃E)

  (∃x) **P**

  | **P(a/x)**

  | **Q**

  Q

Provided that

I. **a** does not occur in an open assumption.

II. **a** does not occur in (∃x) **P**

III. **a** does not occur in **Q**

E.g.: derive (∃x) (Gx ∨ Fx)

1	(∃z)Fz & (∀y) Hy		Ass.
2	(∃z)Fz		1 &E
3		Fb	∀ / ∃E
4		Gb ∨ Fb	3 ∨I
5		(∃x) (Gx ∨ Fx)	4 ∃I
6	(∃x) (Gx ∨ Fx)		2, 3-5 ∃E

SECTION 10.1 THE DERIVATION SYSTEM *PD*

E.g.: derive (∃x) (Gx ∨ Fx)

1	(∃x)Fx & (∀y) Hy		Ass.
2	(∃x)Fx		1 &E
3		Fb	∀ / ∃E
4		Gb ∨ Fb	3 ∨I
5	Gb ∨ Fb		2,3-4 ∃I MISTAKE!
6	(∃x) (Gx ∨ Fx)		5 ∃I

'b' occurs in the sentence we are trying to derive

E.g.: derive (∃x) Fbx

1	(∃x)Fxx		Ass.
2		Fbb	∀ / ∃E
3		(∃x)Fbx	3 ∃I
4	(∃x) Fbx		1, 2-3 ∃E MISTAKE!

‘b’ occurs in **Q**

E.g.: derive (∀x) Fx

1	Gb ⊃ (∀x) Fx		Ass.
2	(∃x) Gx		Ass.
3		Gb	∀ / ∃E
4		(∀x) Fx	1, 3 ⊃E
5	(∀x) Fx		2,3-4 ∃E MISTAKE!

'b' occurs in an open assumption

SECTION 10.1 THE DERIVATION SYSTEM *PD*

- First strategy when using Existential Elimination: When one or more of the currently accessible sentences in a derivation is an existentially quantified sentence, consider using Existential Elimination to obtain the current goal. Assume a substitution instance that contains a constant that does not occur in the existential quantification, in an open assumption, or in the current goal. Work within the Existential Elimination subderivation to derive the current goal.

- Second, when contradictory sentences are available within and Existential Elimination subderivation but cannot be moved out of that subderivation without violating the restrictions, derive another sentence - one that is contradictory to a sentence accessible outside of the Existential Elimination subderivation and one that can be moved out. That sentence will be derivable by the appropriate negation strategy.

E.g.: derive (∃x) Hx

1	(∃z) Gz		Ass.
2	(∀y) (Gy ⊃ Hc)		Ass.
3		Gb	∀ / ∃E
4		Gb ⊃ Hc	2 ∀ E
5		Hc	3,4 ⊃E
6		(∃x) Hx	5 ∃I
7	(∃x) Hx		1,3-6 ∃E

SECTION 10.1 THE DERIVATION SYSTEM *PD*

E.g.: derive (∀x) Fx, ~ (∀x) Fx

1	(∃x) ~Fx	Ass.
2	(∀x) Fx	Ass.
3	~Fa	∀ / ∃E
4	(∀x) Fx	∀ / ~ I
5	Fa	4 ∀E
6	~ Fa	3 R
7	~ (∀x) Fx	4-6 ~I
8	~ (∀x) Fx	1,3-7 ∃E
9	(∀x) Fx	2 R

SECTION 10.1 THE DERIVATION SYSTEM *PD*

- The rules of *PD* now let us define the following syntactic concepts

- *Definition: Derivability in PD*: A sentence **P** of *PL* is *derivable in PD* from a set Γ of sentences if and only if there is a derivation in *PD* in which all the primary assumptions are members of Γ and **P** occurs within the scope of only the primary assumptions.

- *Definition: Validity in PD*: An argument of *PL* is *valid in PD* if and only if the conclusion of the argument is derivable in *PD* from the set consisting of the premises. An argument is *invalid in PD* if and only if it is not valid in *PD*.

SECTION 10.1 THE DERIVATION SYSTEM *PD*

- *Definition: Theorem in PD*: A sentence **P** of *PL* is a *theorem in PD* if and only if **P** is derivable in *PD* from the empty set.

- *Definition: Equivalence in PD*: Sentences **P** and **Q** of *PL* are *equivalent in PD* if and only if **Q** is derivable in *PD* from {**P**} and **P** is derivable in *PD* from {**Q**}.

- *Definition: Inconsistency in PD*: A set Γ of sentences of *PL* is *inconsistent in PD* if and only if there is a sentence **P** suth that both **P** and **~P** are derivable in *PD* from Γ. A set Γ is *consistent in PD* if and only if it is not inconsistent in *PD.*

SECTION 10.3 THE DERIVATION SYSTEM *PD+*

- *PD+* includes all the rules of *PD*, the rules of replacement of *SD+*, and one more unique to *PD+* (*PD+*is no more powerful than *PD,* but proofs are often shorter)
- Here is an example of some replacement rules (we already have) applied correctly to subformulas.

1	(∀x)[(Fx & Hx) ⊃ (∃y)Nxy]	Ass.
2	(∀x)[~ (Fx & Hx) v (∃y)Nxy]	1 Impl
3	(∀x)[~ (Fx & Hx) v ~ ~ (∃y)Nxy]	2 DN
4	(∀x) ~ (Fx & Hx) & ~ (∃y)Nxy]	3 DeM
5	(∀x) ~ [(Hx & Fx) & ~ (∃y)Nxy]	4 Com

- *PD+* includes an additional rule: *Quantifier Negation.*

Where **P** is an open sentence of *PL* in which **x** occurs free, the rule is

Quantifier Negation (QN)

~ (∀x)**P**

(∃x) ~ **P**

or

~ (∃x) **P**

(∀x) ~ **P**

- *PDE* extends *PD* so that our derivation system can work with sentences containing functors and identity. We add two rules for the identity predicate, and then modify our quantifier rules for functors.

Identity Introduction (=I)

$(\forall \mathbf{x})\ \mathbf{x} = \mathbf{x}$

Example: derive a =a

1	$(\forall x)\ x = x$	=I
2	a=a	1 $\forall$E

SECTION 10.4 THE DERIVATION SYSTEM *PDE*

- *Identity Elimination (=E)*

$\mathbf{t}_1 = \mathbf{t}_2$		$\mathbf{t}_1 = \mathbf{t}_2$
P	or	**P**
$\mathbf{P}(\mathbf{t}_1 / / \mathbf{t}_2)$		$\mathbf{P}(\mathbf{t}_2 / / \mathbf{t}_1)$

where t_1 and t_2 are closed terms

Read \` $\mathbf{P}(\mathbf{t}_1 / / \mathbf{t}_2)$' as \`**P** with one or more occurrences of $\mathbf{t_2}$ replaced by $\mathbf{t}_1$.

Remember, closed terms of *PLE* are the individual constants together with the complex terms, such as \`*f*(a,b)'.

Derive Hda

1	c = d	Ass.
2	Hca	Ass.
3	Hda	1,2 =E

Derive $(\forall x)(Fxh \supset Ghx)$

1	h = e	Ass.
2	$(\forall y)(Fye \supset Gey)$	Ass.
3	$(\forall y)(Fyh \supset Ghy)$	1,2 =E
4	$Fah \supset Gha$	3 $\forall$E
5	$(\forall x)(Fxh \supset Ghx)$	4 $\forall$I

SECTION 10.4 THE DERIVATION SYSTEM *PDE*

Derive Wab

1	Haa ⊃ Waa	Ass.
2	Hab	Ass.
3	a = b	Ass.
4	Haa	2,3 =E
5	Waa	1,4 ⊃E
6	Wab	3,5 =E

Derive Hc

1	(∀x)H*f*(a,x)	Ass.
2	c = *f*(a,b)	Ass.
3	*H f*(a,b)	1 ∀E
4	Hc	2,3 =E

SECTION 10.4 THE DERIVATION SYSTEM *PDE*

- The identity predicate is especially useful in symbolizing definite descriptions and arguments that use them.

The roman general who defeated Pompey conquered Gaul.

Julius Caesar is a Roman general, and he defeated Pompey

Julius Caesar conquered Gaul

(∃x) [((Rx & Dxp) & (∀y)[(Ry & Dyp) ⊃ y = x]) & Cxg]

Rj & Djp

Cjg

SECTION 10.4 THE DERIVATION SYSTEM *PDE*

- The identity predicate is especially useful in symbolizing definite descriptions and arguments that use them.

1	(∃x) [((Rx & Dxp) & (∀y)[(Ry & Dyp) ⊃ y = x]) & Cxg]	Ass.
2	Rj & Djp	Ass.
3	((Ra & Dap) & (∀y)[(Ry & Dyp) ⊃ y = a]) & Cag	A/ ∃E
4	(Ra & Dap) & (∀y)[(Ry & Dyp) ⊃ y = a]	3 &E
5	(∀y)[(Ry & Dyp) ⊃ y = a])	4 &E
6	(Rj & Djp) ⊃ j = a	5 ∀E
7	j = a	2,6 ⊃E
8	Cag	3 &E
9	Cjg	1,3-9 ∃E
10	Cjg	

- We need to modify our Existential Introduction and Universal Elimination rules in order to allow for complex terms. The rules are:

Existential Introduction

| P(t/x)
| (∃x)P

where **t** is any closed term

Universal Elimination

| (∀x) P
| P(t/x)

where **t** is any closed term

SECTION 10.4 THE DERIVATION SYSTEM *PDE*

- Consider the following two derivations. Note that in the second derivation we replaced the variable with a complex term `*g*(a)' and not just an individual constant.

Derive: (∃z) Fz

1	(∀y)Fy	Ass.
2	Fa	1 ∀E
3	(∃z) Fz	2 ∃I

Derive: (∃z) F*g*(*z*)

1	(∀y)Fy	Ass.
2	F*g*(a)	1 ∀E
3	(∃z) F*g*(*z*)	2 ∃I

- So for Existential Introduction and Universal Elimination (and only these two) we speak of **instantiating terms** (instead of constants) since sometimes the substitution instance is a constant, and sometimes a closed complex term.

SECTION 10.4 THE DERIVATION SYSTEM *PDE*

- We do not do the same for Existential Elimination and Universal Introduction, i.e., we do not allow substitution instances to be formed from complex terms (we only speak of the instantiating constant). If we modified them as well, we'd make the following mistakes:

Derive: (∀x) Ex

1	(∀x) E*d*(x)	Ass.
2	E *d*(a)	1 ∀E
3	(∀x) Ex	2 ∀I MISTAKE

Derive: (∃x) O*d*(x)

1	(∃x) O(x)		Ass.
2		O*d*(a)	A /∃E
3		(∃x) O*d*(x)	2 ∃I
4	(∃x) O*d*(x)		1, 2-3 ∃E MISTAKE

SECTION 10.4 THE DERIVATION SYSTEM *PDE*

- Here are correct derivations, which are similar to but importantly different than the two we just considered.

Derive: (∀x) E*d*(x)

1	(∀x) Ex	Ass.
2	E*d*(a)	1 ∀E
3	(∀y) E*d*(y)	2 ∀I

Derive: (∃z) F(z)

1	(∃x) F*g*(x)	Ass.
2	F*g*(b)	A /∃E
3	(∃z) F(z)	2 ∃I
4	(∃z) F(z)	1, 2-3 ∃E