3

SUBSTITUTION RULES

If P is a formula, s is a term and x is a variable, then P[s/x] is the formula obtained by substituting s for all free occurrences of x

2-9.
$$(\exists x. P(x, y))[3/y] \equiv \exists x. P(x, 3)$$

 $(\exists x. P(x, y))[3/x] \equiv \exists x. P(x, y)$
NOT free!

If necessary, bound var. in P must be renamed to avoid capture of free var. in s $(\exists x. P(x,y))[f(x)/y] = \exists z. P(z, f(x))$

ASSIGNMENT & SATISFACTION

variables

eg. Consider wff \emptyset : R(f(x), g(y,a)) where $x, y \in \widehat{\mathcal{V}}$ and a is a const. Given interpretation I where

- · domain D is set of integers \mathbb{Z} · $f^{\mathrm{I}} \equiv -$ [subtraction]
- $a^{I} \equiv -5$ $g^{I} \equiv + [addition]$
- · RI = < [Less-than]

and environment $S[x \mapsto 3, y \mapsto 2]$ then under this interpretation & assignment:

$$\phi^{I} \equiv -3 < (2+(-5)) \equiv -3 < -3$$
 is not satisfied!

SATISFACTION & VALIDITY

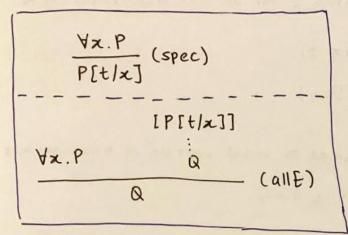
eg. Consider the ff. statement: $\forall x, y. R(x, y) \rightarrow \exists z. R(x, z) \land R(z, y)$ (a). Is it satisfiable?

- → YES! Domain is the real no. and R is interpreted as the < relation
- (b). Is it valid?
 - → NO! Domain? Interpretation of R?

UNIVERSAL QUANTIFICATION EXISTENTIAL QUANTIFICATION

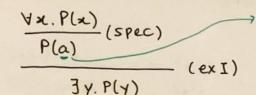
$$\frac{P[x_0/x]}{\forall x.P} (all I)$$

Provided tht. Xo is not tree in the assumptions



$$\frac{P[t/x]}{3x.P} (exI)$$

C.g. Prove tht. Jy. P(y) is true, given tht. Yx. P(x) holds.

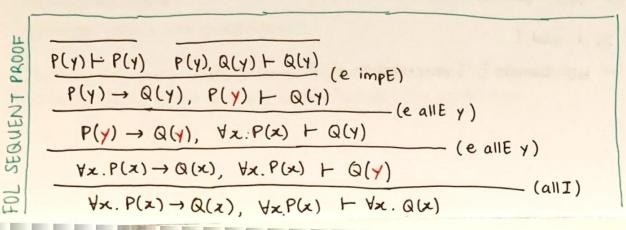


-> We implicitly use the fact tht. our domain is non-empty. It doesn't matter what a is.

eg. Prove tht. Yx. Q(x) is true, given Yx.P(x) and (Yx.P(x) -> Q(x))

		$[P(y) \rightarrow Q(y)]_1$	[P(y)] ₂ (mp)
	\x.P(x)	Q(y)	(all E 2)
$\forall x. P(x) \rightarrow Q(x)$		Q(4)	(all E,)
	Q(y)	NASS A A SECTION	(allI)
LABOR DE LA SERVICIO DE LA CONTRACTOR DE	N 0(1)		

Vx. Q(x)



FOL IN ISABELLE [HOL]

- -> All variables, terms & formulas have types
- The type lang. is built using
 - · base types such as bool and nat
 - · type constructors such as list and set e.g. nat list or nat set
 - function types e.g. nat x nat ⇒ nat which is a ft taking 2 args of type nat
 and return an obj. of type nat
 - · type variables such as 'a, 'b, etc. give rise to polymorphic types 'a ⇒ 'a
- → Consider the predicate a = b mod n
 - definition mod :: "int \Rightarrow int \Rightarrow int \Rightarrow bool"
 where "mod a b n \equiv $\exists k. a = k \neq n + b$ "
- → Isabelle performs type inference, allowing us to write:

 ∀xyn. mod xyn → mod yxn

$$\frac{\Gamma \vdash P[x_0/x]}{\Gamma \vdash \forall x. P} \text{ (allI)} \qquad \frac{\Gamma, P[t/x] \vdash Q}{\Gamma, \forall x. P \vdash Q} \text{ (e allE t)} \qquad \frac{\Gamma, \forall x. P, P[t/x] \vdash Q}{\Gamma, \forall x. P \vdash Q} \text{ (f spec t)}$$

$$\frac{\Gamma \vdash P[t/x]}{\Gamma \vdash \exists x. P} \text{ (r exI t)} \qquad \frac{\Gamma, P[x_0/x] \vdash Q}{\Gamma, \exists x. P \vdash Q} \text{ (e exE)} \qquad \frac{\Gamma, \forall x. \neg P \vdash \bot}{\Gamma \vdash \exists x. P} \text{ (exCIF)}$$

- ► Rule prefixes: e = erule, f = frule, r = rule
- x₀ is some variable not free in hypotheses or conclusion. Isabelle automatically picks fresh names (to ensure soundness!)
- When t suffix is used above (e.g., as in "e allE t"), then the term t can be explicitly specified in Isabelle method using a variant of the existing method. e.g., apply (erule_tac x="t" in allE).
- Rule exCIF is a variation on the standard Isabelle rule exCI introduced in the FOL. thy file on the course webpage. It does not exist as an explicit Isabelle inference rule but can be derived (see FOL. thy).

Why the side conditions on allI and exE?

A (non-)proof of: $\vdash x > 5 \rightarrow \forall x. \ x > 5$:

$$\frac{\overline{x > 5 \vdash x > 5} \quad \text{(assumption)}}{x > 5 \vdash \forall x. \ x > 5} \quad \text{(allI)}$$
$$\frac{+ x > 5 \rightarrow \forall x. \ x > 5}{+ x > 5 \rightarrow \forall x. \ x > 5} \quad \text{(impI)}$$

But it is clearly false that if a particular x is greater than 5, then every x is greater than 5. We have "proven" that x > 5, but not for an **arbitrary** x, only for the **particular** x we had already made an assumption about.

Exercise: Give a non-proof for exE.

Machine assistance: Isabelle keeps track of which variable names are allowed where, so we can only apply the rules in a sound way.