

# CSE4213 Lecture Notes

## Predicate Logic and Substitutions

### Schneider, chapter 2

Computer Science and Software Engineering  
Monash University

20050326 / Lecture 9

Outline

Predicate Calculus

Substitutions

Summary

# Outline

1 Predicate Calculus

2 Substitutions

3 Summary

# Predicates

- **Predicates** are statements which are either true or false
- Predicates can be used to make statements about sets, and hence are very useful in B
- $x \in PRICE$  is a predicate that states something about  $x$ , a potential member of the set  $PRICE$ . If true, then  $x$  is an element of  $PRICE$
- Hence use predicates to define sets with **set comprehension**
- $\{x \mid x \in \mathbb{N} \wedge x \leq 10\}$

# Operations on Predicates

- **negation** (not):  $\neg P$
- **disjunction** (or):  $P \vee Q$
- **conjunction** (and):  $P \wedge Q$
- **implication** (implies):  $P \Rightarrow Q$
- **equivalence** (if and only if):  $P \Leftrightarrow Q$

# Quantification

- We often want to state predicates across a set
- **universal quantification** defines a predicate across all members of a set:  $\forall x.(x \in S \Rightarrow P)$
- **existential quantification** defines a predicate for at least one member of a set:  $\exists x.(x \in S \wedge P)$
- note the difference in connectives (why?)

# Duality of Quantification

- If universal quantification is false, then there must be at least one element that makes it false
- If existential quantification is false, then all elements must not satisfy the predicate.
- Hence we have:

$$\neg \forall x.(x \in S \Rightarrow P) \Leftrightarrow \exists x.(x \in S \wedge \neg P)$$

$$\neg \exists x.(x \in S \wedge P) \Leftrightarrow \forall x.(x \in S \Rightarrow \neg P)$$

# Constraining Predicates

- There are some contexts where it is stated that a predicate  $P$  must **constrain** some list of variables  $z$
- To constrain the variable  $x$ , the predicate  $P$  must contain predicates of the form:  $x \in S$ ,  $x \subseteq S$ ,  $x \subset S$ , or  $x = E$ , where  $x \setminus S$ ,  $x \setminus E$  and  $S$  is a set, and  $E$  is an expression.
- $x \setminus E$  means that  $x$  is not free in  $E$
- We say that  $x$  is a **bound variable** in  $P$ .

# Free Variables

- A variable  $x$  is free in an expression if it is not bound by a quantifier.
- All free instances of  $z$  in  $P$  and  $Q$  are bound in  $\forall z.(P \Rightarrow Q)$  and  $\exists z.(P \wedge Q)$
- Hence,  $z \notin \forall z.(P \Rightarrow Q)$
- Note that  $z$  is **free** in  $P$  and  $Q$ , but **bound** in  $\forall z.(P \Rightarrow Q)$

# Substitutions

## Substitutions as Assignments

- **Substitutions** are crucial to the B-Method
- Substitutions are the way in which we effect change of state
- A substitution (declarative) is the formal equivalent of assignment (imperative)
- $P[E/x]$  means:  $P$ , with every occurrence of  $x$  replaced by  $E$
- multiple substitutions are possible:  $P[E, F, \dots / x, y, \dots]$

# Substitutions

## General Substitution Language

- The substitutions shown in the previous slide are part of the **General Substitution Language** (GSL), a formalism which we will not use much in this unit.
- Instead use the **Abstract Machine Notation**
- Think of substitutions as **Predicate Transformers**

$$[G]P$$

- Example: expanding  $[x := 2]x < 5$  gives  $2 < 5$  (a true predicate)
- Substitutions may or may not **satisfy** a predicate
- *The set of variables  $z$  satisfying the predicate  $P$*  means the variables  $z$  are instantiated to values that when substituted for free instances of  $z$  in  $P$  make the predicate true.

# Substitutions

Philosophy

- The philosophy of program specification may be stated thus:
- Have some starting situation (state) identified by predicate  $Q$
- Want some final situation (state) identified by predicate  $P$
- specify program as a set of substitutions  $G$  such that

$$Q = [G]P$$

# AMN Substitutions

*skip*

**BEGIN G END**

**PRE P THEN G END**

**IF P THEN G [ELSE H] END**

**IF P1 THEN G1 ELSIF P2 THEN G2 ... [ELSE Gn] END**

**CHOICE G OR H END**

**SELECT P THEN G WHEN...**

**WHEN Q THEN H [ELSE I] END**

**CASE E OF EITHER m THEN G OR n THEN H...**

**[ELSE I] END**

**VAR z IN G END**

**ANY z WHERE P THEN G END**

**LET x BE x = E IN G END**

**WHILE P DO G VARIANT E INVARIANT Q END**

*(only in refinements)*

Outline

Predicate Calculus

**Substitutions**

Summary

skip

# Summary

- **Predicates** define properties of B entities
- Predicates define **states of interest**
- **Substitutions** define how values (state) change
- Substitutions allow us to **transform predicates**
- Substitutions build **program specifications** (and ultimately, programs)