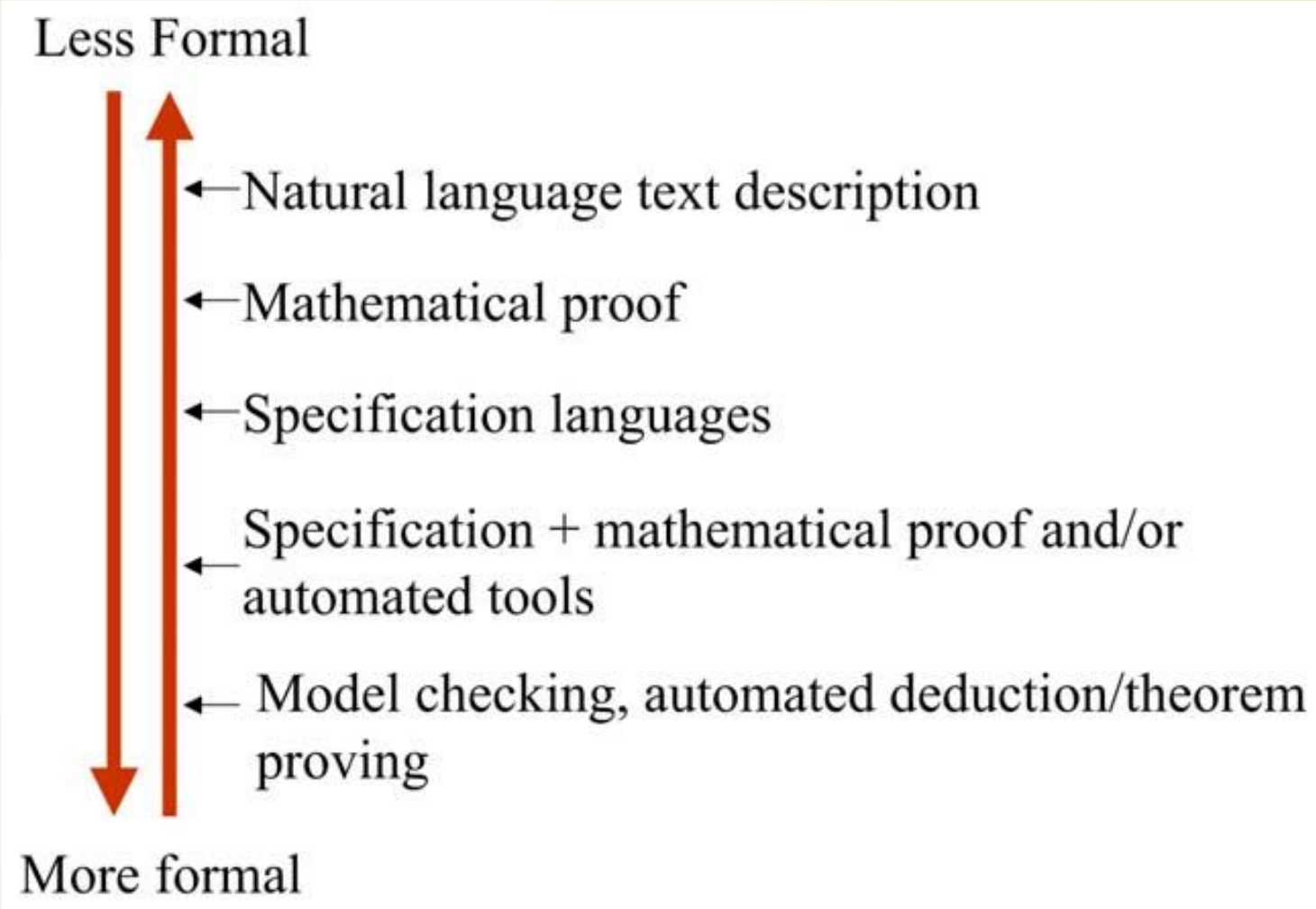
SE4033 Formal Methods for Software Engineering

Formal Methods for Software Engineering

Formalization Spectrum



Logic

Logic or propositional calculus is based on statements, which have truth values
(true or false)

Logic

Symbolic Statement	Translation
$p \vee q$	p or q
$p \wedge q$	p and q
$p \Rightarrow q$	p logically implies q
$p \Leftrightarrow q$	p is logically equivalent to q
$\neg p$ (also $\sim p$)	Not p

Formalization Spectrum

Symbol	Meaning
$\vee$	or
$\wedge$	and
$\neg$	not
$\Rightarrow$	logically implies
$\Leftrightarrow$	logically equivalent
$\forall$	for all
$\exists$	there exists

Quantification

- Quantification is non-logical constants that include names and entities
- All men are mortal

$$\forall X . man(X) \Rightarrow mortal(X)$$

Logic

- Some Examples here

Logic

(a) Negation

p	¬p
T	F
F	T

Logic

(b) Disjunction

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

Logic

(c) Conjunction

p	q	p∧q
T	T	T
T	F	F
F	T	F
F	F	F

Logic

p	q	$p \Rightarrow q$	$\neg p$	$(\neg p) \vee q$
T	T	T	F	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Upper case Roman letters, plus square brackets:
Examples:
H[a] : a is happy
R[a, b] : a respects b
S[a,b,g] : a sold b to g
H[a,b,g,d] : a is happy that b sold g to d

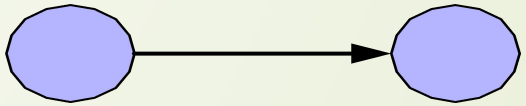
Predicates

Lower case Roman letters, plus parentheses:
Examples:
m(a) : the mother of a
s(a,b) : the sum of a and b
s(a,b,g) : the sum of a, b, g
Variables: lower case Roman letters: z, y, x, …

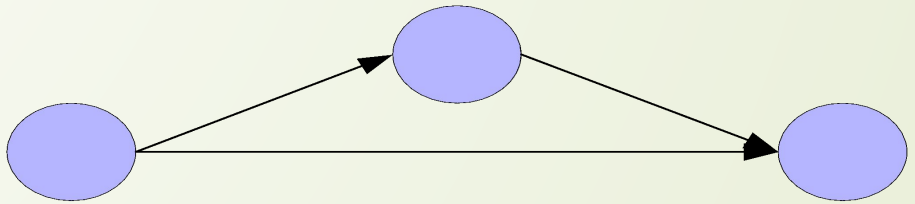
Function Signs

Elementary Logic

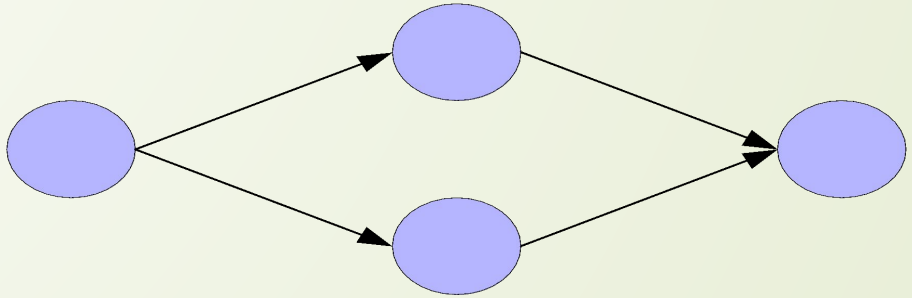
sentences S
noun phrases N
predicates Nk→S
function signs Nk→N
connectives Sk→S
quantifiers V+S→S
description operator V+S→N



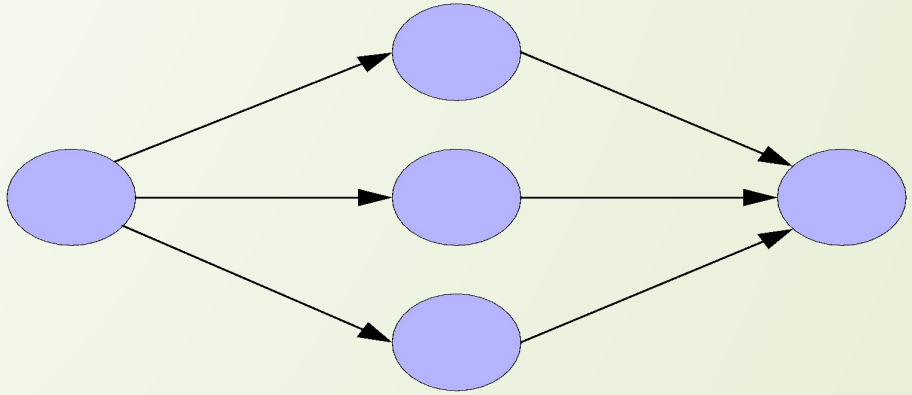
Sequence



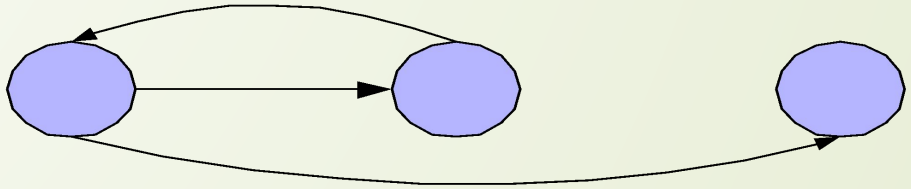
Selection – if statement



Selection – if-else statement



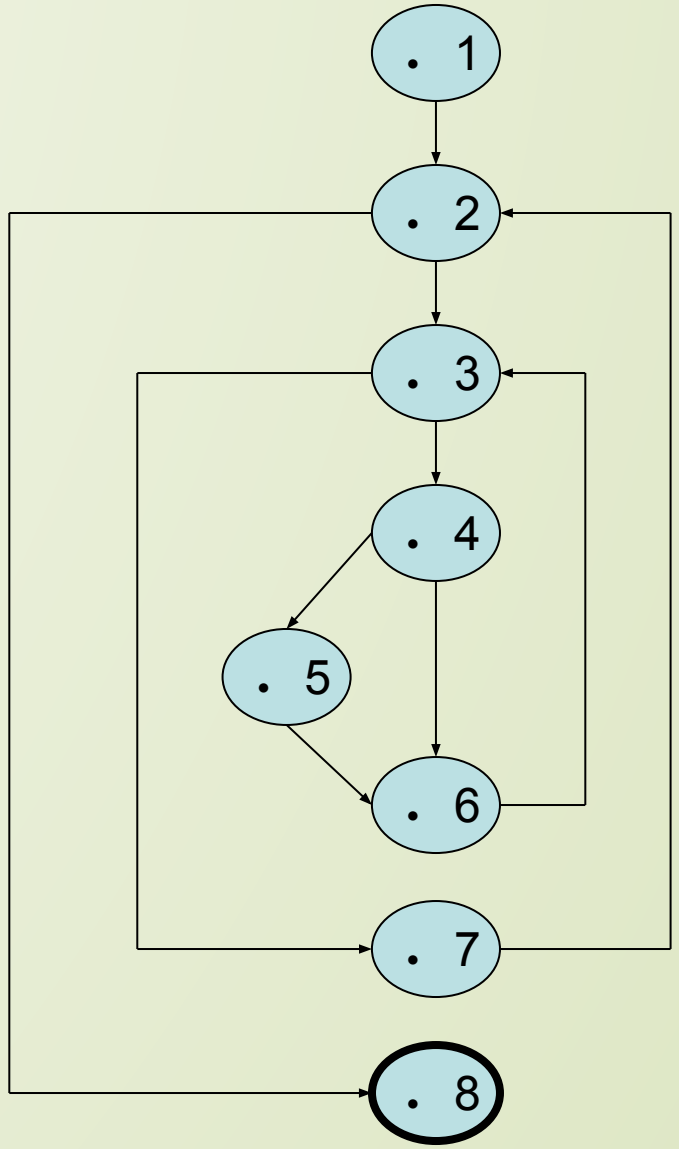
Selection – case statement



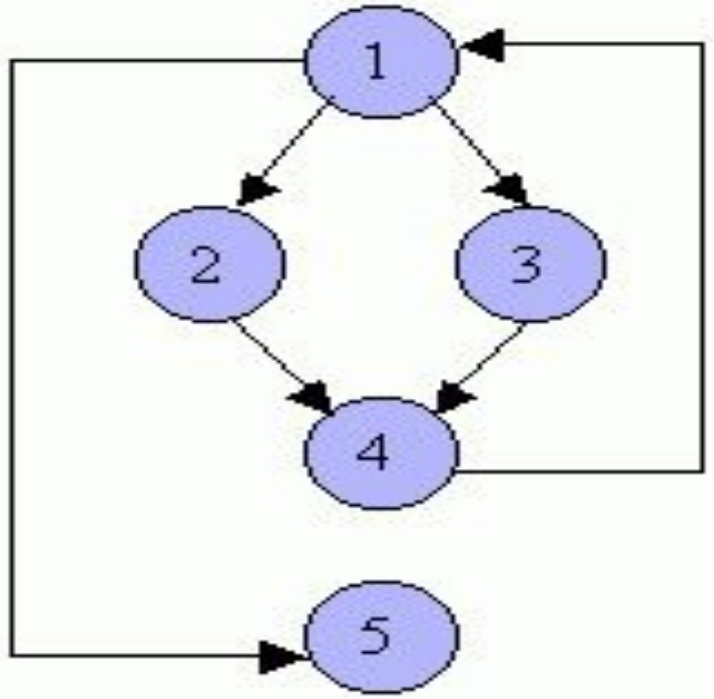
- Loop

Flow graph for bubble sort

```
sorted = false;                 // 1
while (!sorted) {               // 2
    sorted = true;
    for (int i = 0; i < SIZE-1; i++) {    // 3
     if (a[i] > a[i+1]) {       // 4
        swap(a[i], a[i+1]);         // 5
        sorted = false;
     }
    }                           //6
}                               //7
                                //8
```



```
for (i = 0; i < N; i++) {                    //1
  if (condition1)
        // do something here                 //2
  else
        // do something here                 //3
  // something here                         //4
}                                            //5
```



- 2^N Paths