Formal Software Engineering
(génie logiciel avec l'approche formelle)

J. Christian Attiogbé

Master Alma, Septembre 2016
Logics, Sets and Relations

First Order Logic

A proposition is a sentence named P, Q, E... with a value TRUE or FALSE; the construction of a proposition is made with the following grammar:

$$\begin{array}{lll} prop & ::= & P, Q, E, \ldots \\ & | & prop \wedge prop \\ & | & \neg\ prop \\ & | & prop \Rightarrow prop \end{array}$$

Parentheses can be used if necessary.
Other operators (logical connectors) : $\vee$, $\equiv$
The semantics of a proposition (with the connectors) is given by a truth table (Exercice).

Examples of Proposition

$$A\ cat\ with\ a\ hat\ is\ Lion$$

$$Peter\ rides\ bycicle$$

$$0 > 3$$

Inference rules of propositional calculus

$\wedge\ intr$	$\dfrac{HYP \vdash P \qquad HYP \vdash Q}{HYP \vdash P \wedge Q}$	use backward to decompose into simple subgoals with the same hypotheses
$\wedge\ elim$	$\dfrac{HYP \vdash P \wedge Q}{HYP \vdash P \qquad HYP \vdash Q}$	
$\Rightarrow\ intr$	$\dfrac{HYP,\ P \vdash Q}{HYP \vdash P \Rightarrow Q}$	deduction rule
$\Rightarrow\ elim$	$\dfrac{HYP \vdash P \Rightarrow Q}{HYP,\ P \vdash Q}$	anti-deduction

Modus Ponens	$\dfrac{HYP \vdash P \qquad HYP \vdash P \Rightarrow Q}{HYP \vdash Q}$	
Contradiction	$\dfrac{HYP, \neg Q \vdash P \qquad HYP, \neg Q \vdash \neg P}{HYP \vdash Q}$	first rule for $\neg$
	$\dfrac{HYP, Q \vdash P \qquad HYP, Q \vdash \neg P}{HYP \vdash \neg Q}$	second rule for $\neg$

Predicates

Propositional calculus deals with : absolute truth.
Predicate calculus deals with : relative truth,
it is an extension of propositional calculus.

$$x > 2$$

$$x \in \mathbb{N} \Rightarrow x \geq 0$$

Two kinds of variables are used in predicates: free variables and bound variables which are introduced with quantifiers.

How to use predicates

- Substitution

$$[x := 5](x \in \mathbb{N} \Rightarrow x \geq 0)$$

$$(5 \in \mathbb{N} \Rightarrow 5 \geq 0)$$

$$[x := elephant](BigEars(x) \Rightarrow African(x))$$

- Quantification

$$\forall x.BigEars(x) \Longrightarrow African(x),$$

$$\forall x.(Animal(x) \wedge BigEars(x)) \Longrightarrow African(x)$$

How to use predicates

Construction of predicates

$$\begin{array}{lcl} Predicat & ::= & Predicat \Rightarrow Predicat \\ & | & Predicat \wedge Predicat \\ & | & \neg\ Predicat \\ & | & \forall\, Variable.Predicat \\ & | & [Variable := Expression]Predicat \\ & | & Expression = Expression \\ Expression & ::= & Variable \\ & | & [Variable := Expression]Expression \\ Variable & ::= & Identifier \end{array}$$

Usage of Logics

- for modelling : *predicates*

  predicate = formula to be proved

$$P \wedge Q$$

$$P \Rightarrow Q$$

$$0 < 3$$

$$\{0, 3\} \subset \{0, 4, 8, 3\}$$

- for reasoning : *sequents*

$$H \vdash P$$

$$\left.\begin{array}{l} H : Hypotheses \\ P : conjecture \end{array}\right\} predicates$$

Reasoning

- Inference rules

  An inference rule links sequents and its defines a valid step of a proof.
  An inference rule has the following shape:

$$\frac{\sum_1, \sum_2, \ldots, \sum_n}{\sum}$$

  The sequents $\sum_1, \sum_2, \ldots, \sum_n$ are called *antecedents*, and the sequent $\sum$ is called *consequent*.

Reasoning (continued)

- Proof principle
  To prove a sequent, one uses the inference rules
  - as derivation rules : forward rule application,
  - as reduction rules : backward rule application.

Implementation

- Theorem to prove / Inference
  To prove a theorem
  $$P \vdash Q$$
  one transforms it into inference rule
  $$\frac{H \vdash P}{H \vdash Q}$$
- Proof - forward or backward - tactics

Sets and typing

- Predefined Sets (work as types)
  BOOL, CHAR,
  INTEGER ($\mathbb{Z}$), NAT ($\mathbb{N}$), NAT1 ($\mathbb{N}^*$) ,
  STRING
- Cartesian Product E $\times$ F
- The set of subsets (powerset) of E $\mathcal{P}(E)$
  written POW(E)

Set Theory Language

The standard set operators

E, F and T are sets, x an member of F

Description	Notation	Ascii
union	$E \cup F$	`E \/ F`
intersection	$E \cap F$	`E /\ F`
membership	$x \in F$	`x : F`
difference	$E \setminus F$	`E - F`
inclusion	$E \subseteq F$	`E <: F`

+ generalised Union and intersection
+ quantified Union et intersection

Sets and Relations

Set Theory Language

In ascii notation, the negation is written with /.

Description	Notation	Ascii
not member	$x \notin F$	`x /: F`
non inclusion	$E \not\subseteq F$	`E /<: F`
non equality	$E \neq F$	`E /= F`

Generalised Union (à la B)

an operator to achieve the generalised union of well-formed *set expressions*.

$S \in \mathcal{P}(\mathcal{P}(T))$

$\Rightarrow$

$union(S) = \{x \mid x \in T \wedge \exists u.(u \in S \wedge x \in u)\}$

Example

$$union(\{\{aa, ee, ff\},\{bb, cc, gg\},\{dd, ee, uu, cc\}\})$$
$$= \{aa, ee, ff, bb, cc, gg, dd, uu\}$$

Quantified Union

an operator to achieve the quantified union of well-formed *set expressions*.

$\forall x.(x \in S \Rightarrow E \subseteq T)$

$\Rightarrow$

$$\bigcup x.(x \in S \mid E) = \{y \mid y \in T \wedge \exists x.(x \in S \wedge y \in E)\}$$

Exemple

$$UNION(x).(x \in \{1, 2, 3\} \mid \{y \mid y \in NAT \wedge y = x * x\})$$
$$= \{1\} \cup \{4\} \cup \{9\} = \{1, 4, 9\}$$

Generalised Intersection (à la B)

an operator to achieve the generalised intersection of of well-formed *set expressions*.

$S \in \mathcal{P}(\mathcal{P}(T))$

$\Rightarrow$

$inter(S) = \{x \mid x \in T \wedge \forall u.(u \in S \Rightarrow x \in u)\}$

Example

$inter(\{\{aa, ee, ff, cc\}, \{bb, cc, gg\}, \{dd, ee, uu, cc\}\} = \{cc\}$

Quantified Intersection (à la B)

an operator to achieve the quantified intersection of of well-formed *set expressions*.

$\forall x.(x \in S \Rightarrow E \subseteq T)$

$\Rightarrow$

$\bigcap x.(x \in S \mid E)$

$= \{y \mid y \in T \wedge \forall x.(x \in S \Rightarrow y \in E)\}$

Example

$INTER(x).(x \in \{1, 2, 3, 4\} \mid \{y \mid y \in \{1, 2, 3, 4, 5\} \wedge y > x\})$

$= inter(\{\{1, 2, 3, 4, 5\}, \{2, 3, 4, 5\}, \{3, 4, 5\}, \{4, 5\}\})$

Relations

Description	Notation	Ascii
relation	$r : S \leftrightarrow T$	r : S <-> T
domain	$dom(r) \subseteq S$	dom(r) <: S
range	$ran(r) \subseteq T$	ran(r) <: T
composition	$r;s$	r;s
composition r(s)	$r \circ s$	r(s)
identity	$id(S)$	id(S)

Relations (continued)

Description	Notation	Ascii
domain restrictition	$S \lhd r$	S <| r
range restriction	$r \rhd T$	r |> T
domain antirestriction	$S \lhd r$	S <<| r
range antirestriction	$r \rhd T$	r |>> T
inverse	$r^{\sim}$	r ~
relationnelle image	$r[S]$	r[S]
overiding	$r1 \oplus r2$	r1 <+ r2
direct product of rel.	$r1 \otimes r2$	r1 >< r2
closure	$closure(r)$	closure(r)
reflexive trans. closure	$closure1(r)$	closure1(r)

Functions

Description	Notation	Ascii
partial function	$S \pfun T$	S +-> T
total function	$S \fun T$	S --> T
partial injection	$S \pinj T$	S >+-> T
total injection	$S \inj T$	S >-> T
partial surjection	$S \psur T$	S +->> T
total surjection	$S \sur T$	S -->> T
total bijection	$S \bij T$	S >->> T
lambda abstraction	$\%x.(P \mid E)$	