First Order Predicate Logic (FOL) – Formulas

Let $\Sigma = (S, \Omega)$ be a signature.

$PL(\Sigma)$ is the smallest set with

(i) $t = u \in PL(\Sigma)$,
if X set of variables for Σ, $s \in S$, $t, u \in T_{\Sigma(X),s}$

(ii) $(\varphi_1 \wedge \varphi_2) \in PL(\Sigma)$ if $\varphi_1, \varphi_2 \in PL(\Sigma)$,

(iii) $(\neg\varphi) \in PL(\Sigma)$ if $\varphi \in PL(\Sigma)$,

(iv) $(\forall x : s \,.\, \varphi) \in PL(\Sigma)$ if
$s \in S$, x variable of sort s, and $\varphi \in PL(\Sigma)$.

FOL – Classical Abbreviations

$(\varphi_1 \vee \varphi_2)$	for	$(\neg((\neg\varphi_1) \wedge (\neg\varphi_2)))$;
$(\varphi_1 \Rightarrow \varphi_2)$	for	$((\neg\varphi_1) \vee \varphi_2)$;
$(\varphi_1 \Leftrightarrow \varphi_2)$	for	$((\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1))$;
$(\exists x : s \,.\, \varphi)$	for	$(\neg(\forall x : s \,.\, (\neg\varphi)))$;
$(\forall x_1 : s_1, \ldots, x_k : s_k \,.\, \varphi)$	for	$(\forall x_1 : s_1 \,.\, (\ldots (\forall x_k : s_k \,.\, \varphi)))$;
$(\exists x_1 : s_1, \ldots, x_k : s_k \,.\, \varphi)$	for	$(\exists x_1 : s_1 \,.\, (\ldots (\exists x_k : s_k \,.\, \varphi)))$;
T	for	$(\psi \Leftrightarrow \psi)$;
F	for	$(\neg T)$.

$\varphi_1, \varphi_2, \varphi_3$ arbitrary formulas,

ψ : a fixed but not further specified formula.

Conditional Terms in CASL

$T1, T2$: Terms

F : Formula

conditional term in CASL: $T1\ when\ F\ else\ T2$

The enclosing atomic formula

$$`A[T1\ when\ F\ else\ T2]'$$

expands to

$$`(A[T1]\ if\ F) \wedge (A[T2]\ if\ \neg F)'.$$

Free Variables

Let

Σ be a signature,

$\varphi \in PL(\Sigma)$ be a formula.

The set $free(\varphi)$ is defined by induction on the structure of φ :

(i)	$free(t = u)$	$:=$	$Var(t) \cup Var(u)$
(ii)	$free(\varphi_1 \wedge \varphi_2)$	$:=$	$free(\varphi_1) \cup free(\varphi_2)$
(iii)	$free(\neg(\varphi))$	$:=$	$free(\varphi)$
(iv)	$free(\forall x : s.\varphi)$	$:=$	$free(\varphi) \backslash \{x\}$

An easy Notation

Let $f : A \to B$ be a function,
a, b elements ($a \in A, b \in B$ allowed, but *not requiered!*)

$$f[b/a] := \begin{cases} A \cup \{a\} & \to & B \cup \{b\} \\ x & \mapsto & \begin{cases} b; & x = a \\ f(x); & else \end{cases} \end{cases}$$

Semantic of Formulas

Let Σ be a signature,
A be a Σ-algebra,
$\varphi \in PL(\Sigma)$ be a formula, and
$\alpha : X \to A$ be an assignment with $X \supseteq free(\varphi)$.

The *value* $A(\alpha)(\varphi) \in \{true, false\}$ is defined by induction on the structure of φ :

(i) $A(\alpha)(t = u) = true$ iff $A(\alpha)(t) = A(\alpha)(u)$.

(ii) $A(\alpha)(\varphi_1 \wedge \varphi_2) = true$ iff $A(\alpha)(\varphi_1) = true$ and $A(\alpha)(\varphi_2)$

(iii) $A(\alpha)(not(\varphi)) = true$ iff $A(\alpha)(\varphi) = false$.

(iv) $A(\alpha)(\forall x : s.\varphi)) = true$ iff $A(\alpha[a/x])(\varphi) = true$ for all $a \in A(s)$.

FOL – Satisfaction Relation

Let $\Sigma = (S, \Omega)$ be a signature.

$$A \models \varphi \text{ iff}$$
$$A(\alpha)(\varphi) = true \text{ for all assignments } \alpha : free(\varphi) \to A.$$

CASL-Syntax for FOL

```
BASIC-ITEMS::= SIG-ITEMS
             | forall VAR-DECL ;...; VAR-DECL
                 "." FORMULA "."..."." FORMULA ;/
SIG-ITEMS   ::= sort/sorts SORT-ITEM;...;SORT-ITEM;/
             |  op/ops OP-ITEM ;...; OP-ITEM ;/
SORT-ITEM   ::= SORT ,..., SORT
OP-ITEM     ::= OP-NAME ,..., OP-NAME : OP-TYPE
OP-TYPE     ::= SOME-SORTS  -> SORT  |  SORT
SOME-SORTS  ::= SORT *...* SORT
VAR-DECL    ::= VAR ,..., VAR : SORT
```

```
FORMULA ::= TERM = TERM
          | ( FORMULA )
          | FORMULA /\ FORMULA /\ ... /\ FORMULA
          | FORMULA => FORMULA
          | FORMULA if FORMULA
          | QUANTIFIER  VAR-DECL ;...; VAR-DECL
                "." FORMULA
          | FORMULA \/ FORMULA \/...\/ FORMULA
          | FORMULA <=> FORMULA
          | not FORMULA
          | true | false
          | ...
```

spec BOOLEAN =

sort $Bool$;

ops $TRUE, FALSE : Bool$;

$NOT : Bool \to Bool$;

$\_\_AND\_\_ : Bool \times Bool \to Bool$;

forall $x, y : Bool$

- $x = TRUE \vee x = FALSE$ %(no_junk)%
- $\neg\ TRUE = FALSE$ %(no_confusion)%
- $NOT(TRUE) = FALSE$ %(NOT_TRUE)%
- $NOT(FALSE) = TRUE$ %(NOT_FALSE)%
- $x\ AND\ y = y\ when\ x = TRUE\ else\ FALSE$ %(AND)%

end

spec DATABASE =

sort $Database; String; Nat$

ops $initial : Database;$

$0 : Nat;$

$look\_up : Database \times String \to Nat;$

$update : Database \times String \times Nat \to Database$

forall $s : Database; n : Nat; v, w : String$

- $look\_up(initial, v) = 0$ %(initial)%
- $v = w \Rightarrow$ %(l_up_1)%

  $look\_up(update(s, v, n), w) = n$
- $\neg\ v = w \Rightarrow$ %(l_up_2)%

  $look\_up(update(s, v, n), w) = look\_up(s, w)$

end

%prec $\{\_\_ + \_\_, \_\_ - \_\_\} < \{\_\_ * \_\_, \_\_\ div\ \_\_, \_\_\ mod\ \_\_\}$

spec $\textsc{Nat} =$

sort $Bool;$

ops $TRUE, FALSE\ :\ Bool;$

sort $Nat;$

ops $0 : Nat;$

$suc : Nat \to Nat;$

$\_\_ < \_\_ : Nat \times Nat \to Bool;$

forall $n, m : Nat$

- $0 < suc(n) = TRUE$
- $suc(n) < 0 = FALSE$
- $suc(m) < suc(n) = m < n$

ops $\_\_ + \_\_, \_\_ - \_\_, \_\_ * \_\_,$
$\_\_div\_\_, \_\_mod\_\_ : Nat \times Nat \rightarrow Nat;$

forall $n, m, r, s : Nat$

- $0 + n = n$
- $suc(m) + n = suc(m + n)$
- $m - n = r \Leftrightarrow m = r + n$
- $0 * n = 0$
- $suc(m) * n = m * n + n$
- $m \ div \ n = r \Leftrightarrow (\exists s : Nat \bullet m = n * r + s \wedge s < n)$
- $m \ mod \ n = s \Leftrightarrow (\exists r : Nat \bullet m = n * r + s \wedge s < n)$

end

spec $\textsc{ArrayOfNat} = \textsc{Nat}$ **then**

sort $Array;$

ops $maxIndex, error : Nat;$

$initial : Array;$

$\_\_!\_\_ : Array \times Nat \to Nat;$

$\_\_!\_\_ := \_\_ : Array \times Nat \times Nat \to Array;$

forall $A : Array; i, j, e : Nat$

- $initial!j = error$
- $j <= maxIndex = FALSE \Rightarrow A!j = error$
- $j <= maxIndex = TRUE \wedge j = i \Rightarrow (A!i := e)!j = e$
- $j <= maxIndex = TRUE \wedge \neg\, j = i \Rightarrow (A!i := e)!j = A!j$

spec B1 =
  sort $Bool$;
  ops $TRUE, FALSE : Bool$; $\_\_AND\_\_ : Bool \times Bool \to Bool$;
  forall $x, y : Bool$
  - $x = TRUE \lor x = FALSE$
  - $\neg\ TRUE = FALSE$

  forall $x, y : Bool$
  - $x\ AND\ y = y\ when\ x = TRUE\ else\ FALSE$

end

spec B2 =
  . . .
  forall $x, y : Bool$
  - $TRUE\ AND\ TRUE = TRUE$
  - $TRUE\ AND\ FALSE = FALSE$
  - $FALSE\ AND\ TRUE = FALSE$
  - $FALSE\ AND\ FALSE = FALSE$

end

spec B3 =
  . . .
  forall $x, y : Bool$
  - $x\ AND\ y = y\ AND\ x$
  - $TRUE\ AND\ y = y$
  - $FALSE\ AND\ y = FALSE$

end

Polymorphic Specifications

spec $\textsc{Automata} = \textsc{Boolean}$ **then**

sort $State, Alphabet$

ops $initial : State;$

$\_\_ - - \_\_ - - > \_\_ : State * Alphabet * State \to Bool$

end

spec $\textsc{DirectedGraph} = \textsc{Boolean}$ **then**
sort $Node$
op $\_\_connectedWith\_\_ : Node * Node \to Bool$
end

spec $\textsc{UnDirectedGraph} = \textsc{DirectedGraph}$ **then**
forall $n, m : Node$
- $n\ connectedWith\ m\ =\ m\ connectedWith\ n$

end