# Formal Methods Formulas and Algorithm

Nicola Di Santo

July 2020

## 1 Introduciton

this document is a collection of formulas to use during formal methods exercises. The .tex file is available to make it always richer, bigger and better.

# 2 FOL

**$\alpha$-rules**:
$\frac{\phi\wedge\psi}{\begin{array}{c}\phi\\\psi\end{array}}$ $\frac{\neg(\phi\vee\psi)}{\begin{array}{c}\neg\phi\\\neg\psi\end{array}}$ $\frac{\neg(\phi\supset\psi)}{\begin{array}{c}\phi\\\neg\psi\end{array}}$ $\frac{\neg\neg\phi}{\phi}$

**$\beta$-rules**:
$\frac{\phi\vee\psi}{\phi|\psi}$ $\frac{\neg(\phi\wedge\psi)}{\neg\phi|\neg\psi}$ $\frac{\phi\supset\psi}{\neg\phi|\psi}$

**extra-rules**:
$\frac{\begin{array}{c}\phi\\\neg\phi\end{array}}{X}$ $\frac{\phi\equiv\psi)}{\begin{array}{c}\phi|\neg\phi\\\psi|\neg\psi\end{array}}$ $\frac{\neg(\phi\equiv\psi)}{\begin{array}{c}\neg\phi|\ \phi\\\psi\ |\neg\psi\end{array}}$

**$\delta$-rules**:
$\frac{\forall x.\phi(x)}{\phi(t)}$ $\frac{\neg\exists x.\phi(x)}{\neg\phi(t)}$

**$\gamma$-rules**:
$\frac{\neg\forall x.\phi(x)}{\neg\phi(c)}$ $\frac{\exists x.\phi(x)}{\phi(c)}$

**Tableaux**:
prove $\phi \equiv \psi$: check $\neg(\phi \equiv \psi)$ is UNSAT
$\phi$ is valid: $\neg\phi$ is UNSAT
$\Gamma \models \phi$: check $\Gamma \cup \neg\phi$ closes (UNSAT)
$\Gamma$ SAT: check for an open branch of $\Gamma$
In general each existential is a fresh new constant while universal can be any term also the constant of the existential. To clash instantiate existential and then use universal to make the clash.

# 3 UML TO FOL

a is the attribute name,A the association name, C is the class name, T is the type name, i and j the multiplicities, P is type name for parameter, R is type name for return value. Class:
$\forall x, y.a(x, y)\supset C(x) \wedge T(y)$
$\forall x.C(x)\supset i \leq \{y|a(x, y)\} \leq j$

Association:
$\forall x_1, ..., x_n.A(x_1, ..., x_n)\supset C(x_1) \wedge ...C(x_n)$
$\forall x_1.C(x_1)\supset i \leq \{x2|a(x_1, x_2, ...)\} \leq j$
$\forall x_2.C(x_2)\supset i \leq \{x1|a(x_1, x_2, ...)\} \leq j$

Generalization:
$\forall x.C_i(x)\supset C(x) for i = 1, ..., n$ (is-a)
$\forall x.C_i(x)\supset\neg C_j(x) for i \neq j$ (disjoint)
$\forall x.C(x)\supset C_1 \vee .. \vee C_i(x) ... \vee C_n(x)$ (completeness)

Subset:
$\forall x, y.assoc_1(x, y)\supset assoc_2(x, y)$ can refine mult.

Association Class:
$\forall x, y, z.a(x, y, z)\supset A(x, y) \wedge T(z)$
$\forall x, y A(x, y)\supset\forall z.i \leq \{z|a(x, y, z)\} \leq j$ + assoc mult.

Reification (when assoc is a class+key constr):
$\forall x, y r_i(x, y)\supset A(x) \wedge C_i(y) for i = 1, .., n$
$\forall x A(x)\supset\exists y.r_i(x, y) for i = 1, .., n$
$\forall x, y, y' r_i(x, y) \wedge r_i(x, y')\supset y = y' for i = 1, .., n$
$\forall y_1, ..., y_n, x, x'. \wedge_{i=1}^{n} r_i(x, y_i) \wedge r_i(x', y_i)\supset x = x'$

Methods:
$\forall x, p_1, ..., p_m, r.f_{C,P_1,...,P_m}(x, p_1, ..., p_m, r)\supset C(x) \wedge (\wedge_{i=i}^{m} P_i(p_i)) \wedge R(r)$
$\forall x, p_1, ..., p_m, r, r'.f_{C,P_1,...,P_m}(x, p_1, ..., p_m, r) \wedge f_{C,P_1,...,P_m}(x, p_1, ..., p_m, r')\supset r = r'$

# 4 Evaluation Semantic

$\frac{(a,s)\rightarrow s'}{true}$ if $s \models Pre(a) \wedge s' \models Post(a, s)$

$\frac{(skip,s)\rightarrow s}{true}$

$\frac{(\delta_1;\delta_2,s_0)\rightarrow s_f}{(\delta_1,s_0)\rightarrow s_1\wedge(\delta_2,s_1)\rightarrow s_f}$

$\frac{(if(\phi)then\{\delta_1\}else\{\delta_2\},s)\rightarrow s'}{(\delta_1,s)->s'}, s \models \phi$

$\frac{(if(\phi)then\{\delta_1\}else\{\delta_2\},s)\rightarrow s'}{(\delta_2,s)->s'}, s \not\models \phi$

$\frac{(while(\phi)do\{\delta\},s)\rightarrow s'}{(\delta,s)\rightarrow s''\wedge(while\phi do\{\delta\},s'')\rightarrow s'}, s \models \phi$

$\frac{(while(\phi)do\{\delta\},s)\rightarrow s'}{true}, s \not\models \phi$

# 5 Transition Semantic

## 5.1 Transition Rules

$\frac{(a,s)\rightarrow(\epsilon,s')}{true}$ if $s \models Pre(a) \wedge s' \models Post(a, s)$

$\frac{(skip,s)\rightarrow(\epsilon,s')}{true}$

$\frac{(\delta_1;\delta_2,s)\rightarrow(\delta_1';\delta_2,s')}{(\delta_1,s)\rightarrow(\delta_1',s')}$

$\frac{(\delta_1;\delta_2,s)\rightarrow(\delta_2',s')}{(\delta_2,s)\rightarrow(\delta_2',s')}$ if $(\delta_1, s)\checkmark$

$\frac{(if(phi)then\{\delta_1\}else\{\delta_2\},s)\rightarrow(\delta_1',s')}{(\delta_1,s)\rightarrow(\delta_1',s')}$ if $s \models \phi$

$\frac{(if(phi)then\{\delta_1\}else\{\delta_2\},s)\rightarrow(\delta_2',s')}{(\delta_2,s)\rightarrow(\delta_2',s')}$ if $s \not\models \phi$

$\frac{(while(\phi)do\{\delta\},s)\rightarrow(\delta';while(\phi)do\{\delta\},s')}{\delta,s)\rightarrow(\delta',s')} if s \models \phi$

## 5.2 Termination Rules

$\frac{(\epsilon,s)\checkmark}{true}$

$\frac{(\delta_1;\delta_2,s)\checkmark}{(\delta_1,s)\checkmark\wedge(\delta_2,s)\checkmark}$

$\frac{(if(phi)then\{\delta_1\}else\{\delta_2\},s)\checkmark}{(\delta_1,s)\checkmark}$

$\frac{(if(phi)then\{\delta_1\}else\{\delta_2\},s)\checkmark}{(\delta_2,s)\checkmark}$

$\frac{(while(\phi)do\{\delta\},s)\checkmark}{true}, if s \models \neg\phi$

$\frac{(while(\phi)do\{\delta\},s)\checkmark}{(\delta,s)\checkmark}, if s \models \phi$

# 6 Hoare Logic

$P \Rightarrow I$
$(\neg g \wedge I) \Rightarrow Q$
$\{g \wedge I\}S\{I\}$: find wp of S and check if $(g \wedge I) \Rightarrow wp$

# 7 CTL TO $\mu$-CALC

$EXp :< - > p$
$AXp : [-]p$
$EFp : \mu X.p \vee < - > X$
$AFp : \mu X.p \vee [-]X$
$pEUq : \mu X.q \vee p \wedge < - > q$
$pAUq : \mu X.q \vee p \wedge [-]q$
$EGp : \nu X.p \wedge < - > X$
$AGp : \nu X.p \wedge [-]X$

# 8 Conjunctive Queries

**Algorithm 1:** Canonical Interpretation

**Input:** q: conj query

$\Delta^{I_q}$ = all constant and variable of q ;
$P^{I_q} = (t_1, ..., t_n)$, ...for all $P_i(t_1, ..., t_n)$ in q;
$c^{I_q} = c$ c is in q ;

**Algorithm 2:** $I \models q$ $(\exists h(I_q) = I)$

**Input:** q: conj query
I: interpretation (DB)

write q in canonical Interp.;
find assignment $\alpha(.)$ for each variable in q;
assign to each constant itself: $\alpha(c) = c$ ;
**return** $\alpha$ as the homomorphism between I and $I_q$ if exist or $I \not\models q$;

**Algorithm 3:** $q_1 \subseteq q_2$

**Input:** $q_1$: conj query
$q_2$: conj query
show $q_1 \subseteq q_2$i.e. $q_2 \Rightarrow q_1$

**begin** containement
  freeze all variable by assigning a constant: assume $q_1(x, y) \leftarrow e(x, y, z)$... you must freeze only the one in the argument $q_1(c_1, c_2) \leftarrow e(c_1, c_2, z)$... ;
  build $I_{q_1}$;
  check if db tables of $I_{q_1}$ are true in $q_2$: find an assignment to $q_2$ variables that makes $I_{q_1} \models q_2$ by guessing ;
**end**
**begin** homomorphism
  `/* To verify the homomorphism check` $I_{q_1} \models q2$ `iff` $\exists h.I_{q_2} \Rightarrow I_{q_1}$ `*/`
  compute $I_{q_2}$ as interpretation of $q_2$ ;
  compute mapping from object o$I_{q_2}$ to $\delta^{I_{q_1}}$ ;
  check if mapping holds also for tuples ;
**end**

**Algorithm 4:** Check if incomplete db $\models q$

**Input:** q: conj query
Incomplete DB

transform DB into $q_D$ where each null become an existentially quantified variable ;
DB$\models q$ iff $q_D \subseteq q$ (see containment algorithm);

# 9 Bisimilarity

A state $s_0$ of transition system S is bisimilar, or simply equivalent, to a state $t_0$ of transition system T iff there exists a bisimulation between the initial states $s_0$ and $t_0$ (note: bisimilarity the largest bisimulation).
A binary relation R is a bisimulation if $(s, t) \in R$ implies that s is final iff t is final and for all action a if $s \rightarrow_a s'$ then $\exists t'.t \rightarrow_a t' and(s', t') \in R$ if $t \rightarrow_a t'$ then $\exists s'.s \rightarrow_a s' and(s', t') \in R$

**Algorithm 5:** Check Bisimulation between T and S

**Input:** T, S

Compute R = $T \times S$ ;
remove from R all tuples (s,t) where s is final and t is not and vice versa ;
remove from R all tuples (s,t) where s can do $a_i$ and t cannot (and vice versa);
remove from R all tuples $(s_i, t_j)$ that can reach (s',t') but then (s',t') is not in R ;
**return** R;