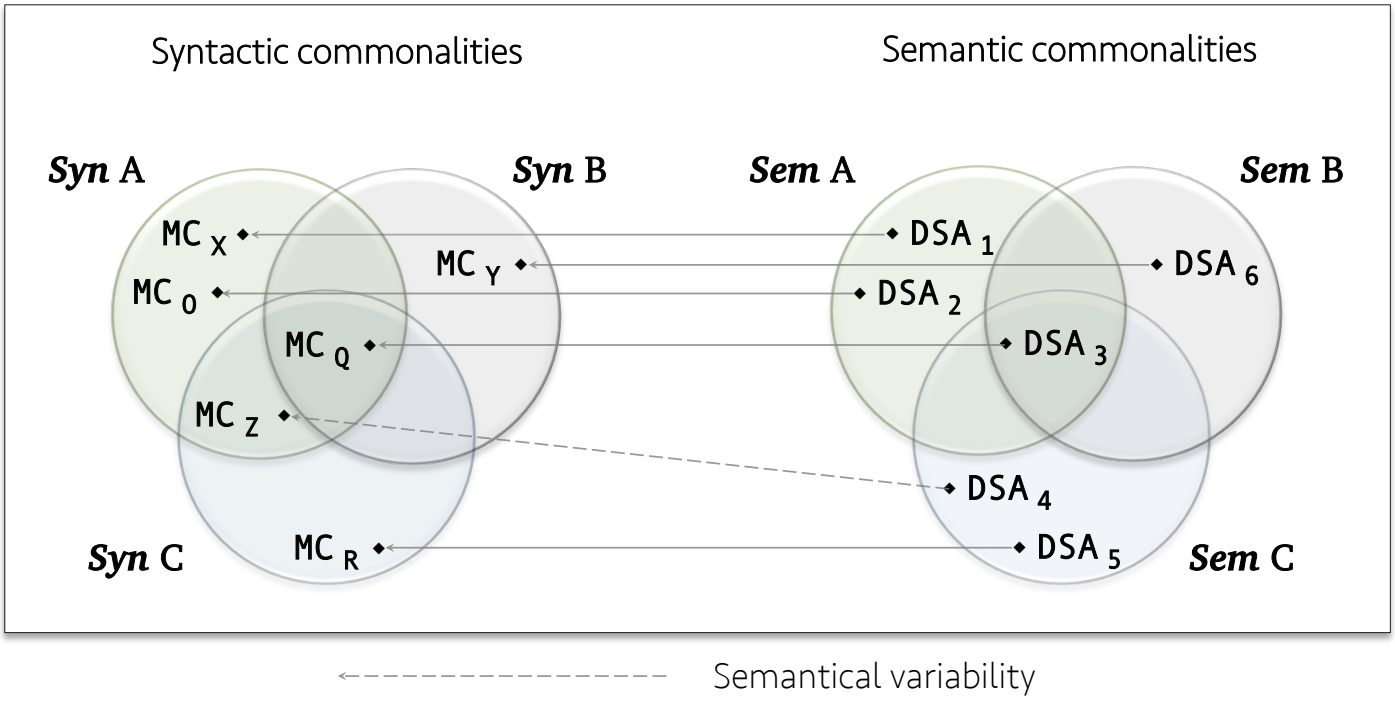
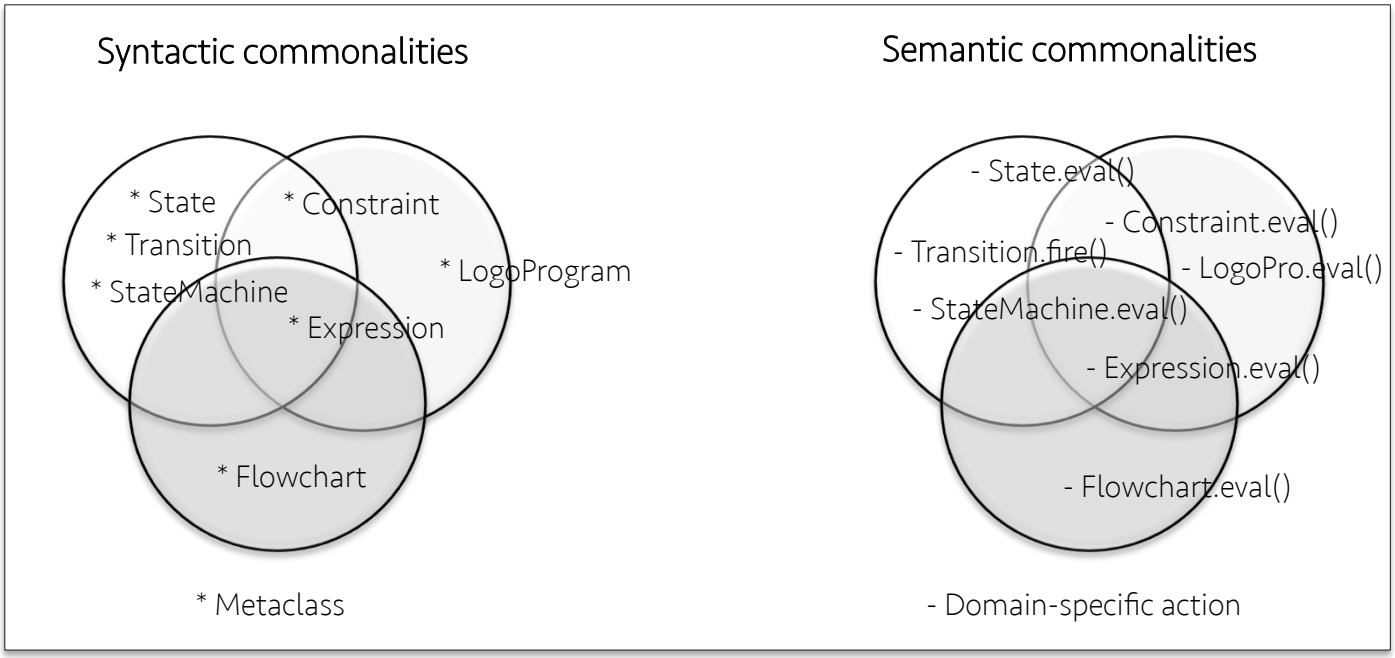
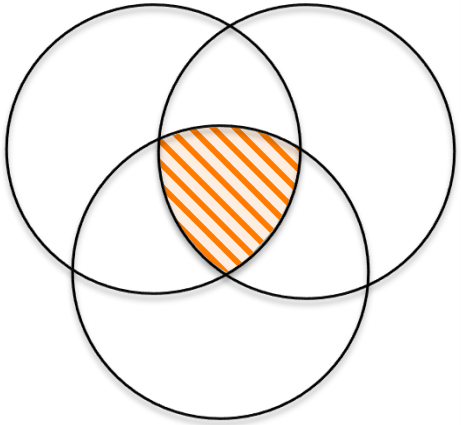
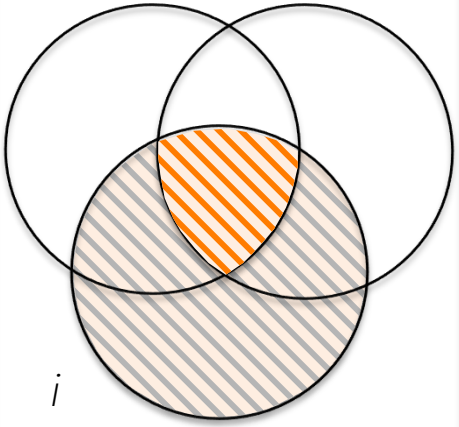
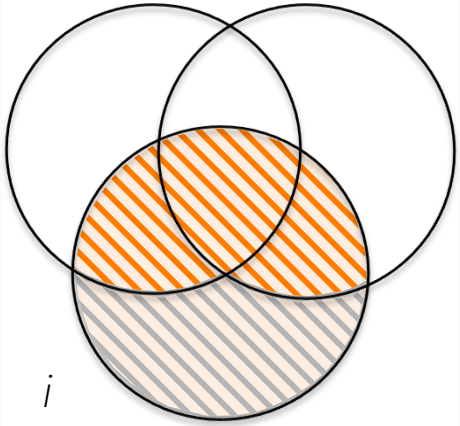
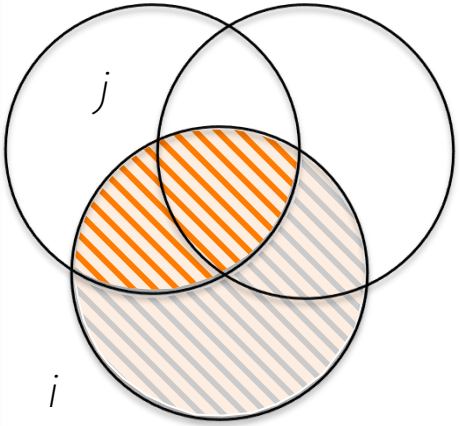
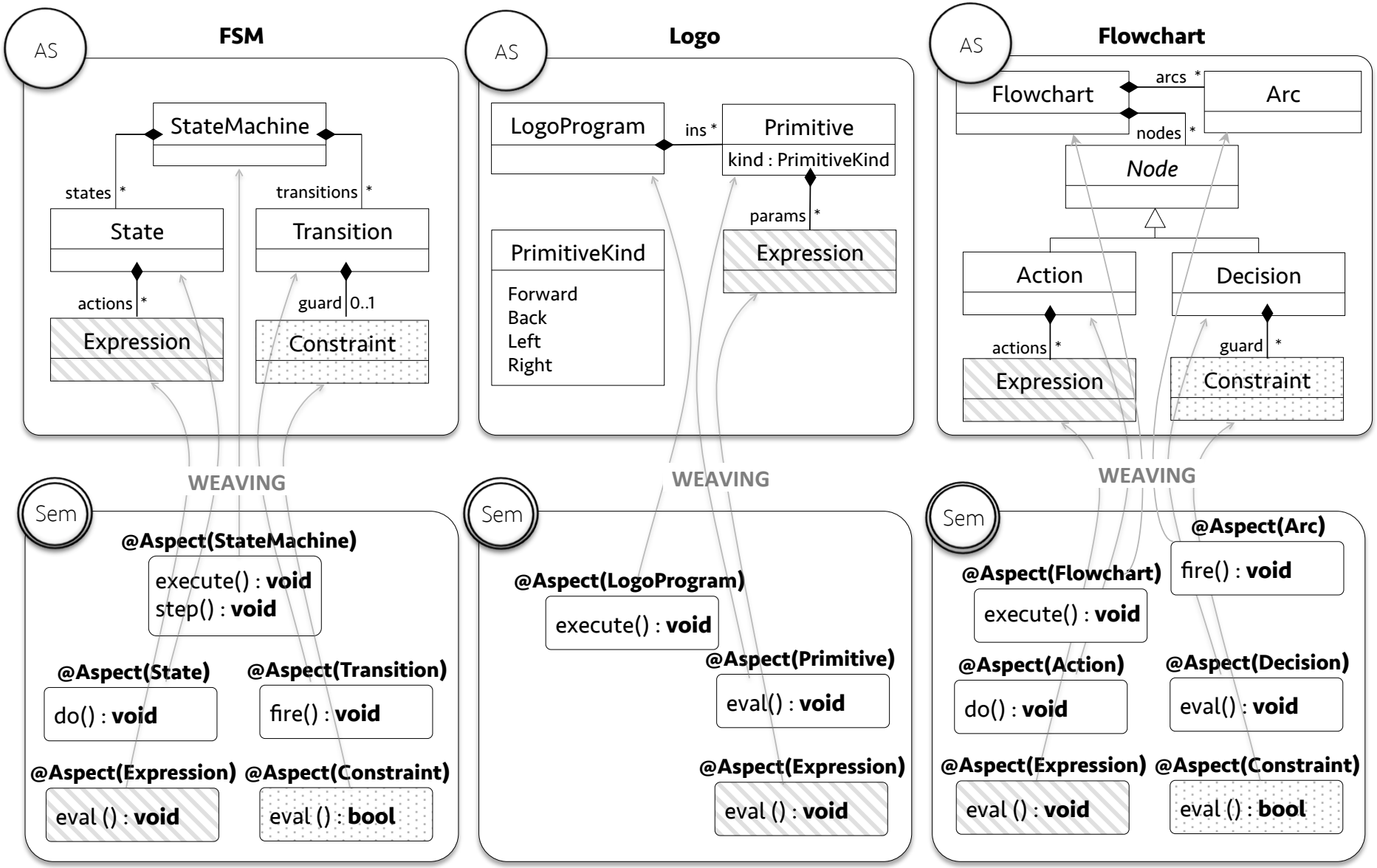
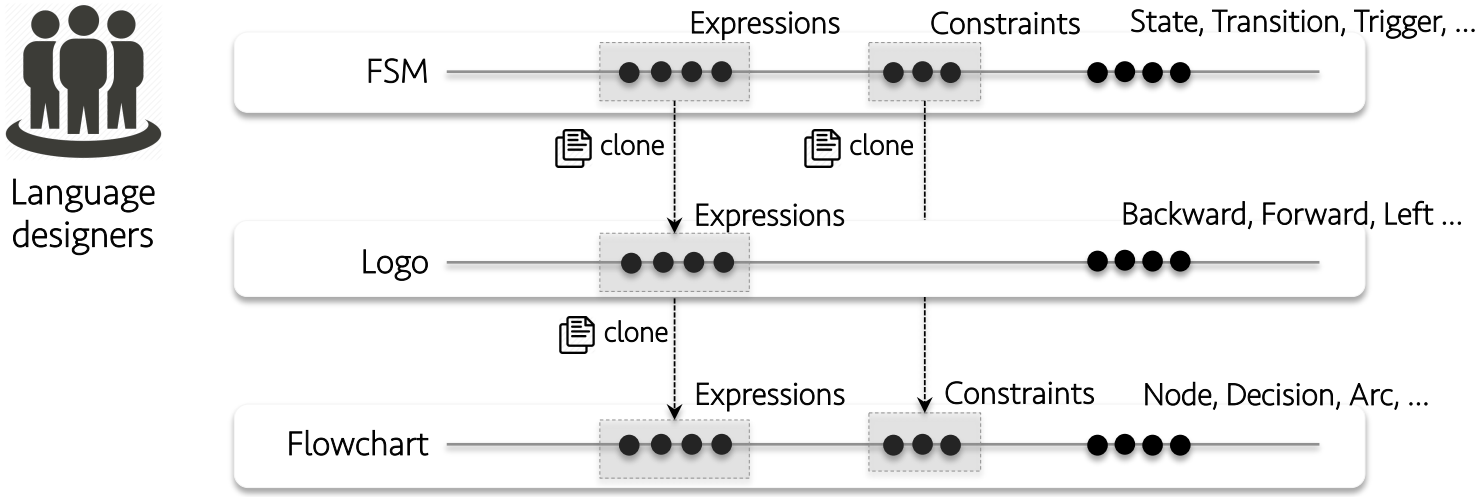
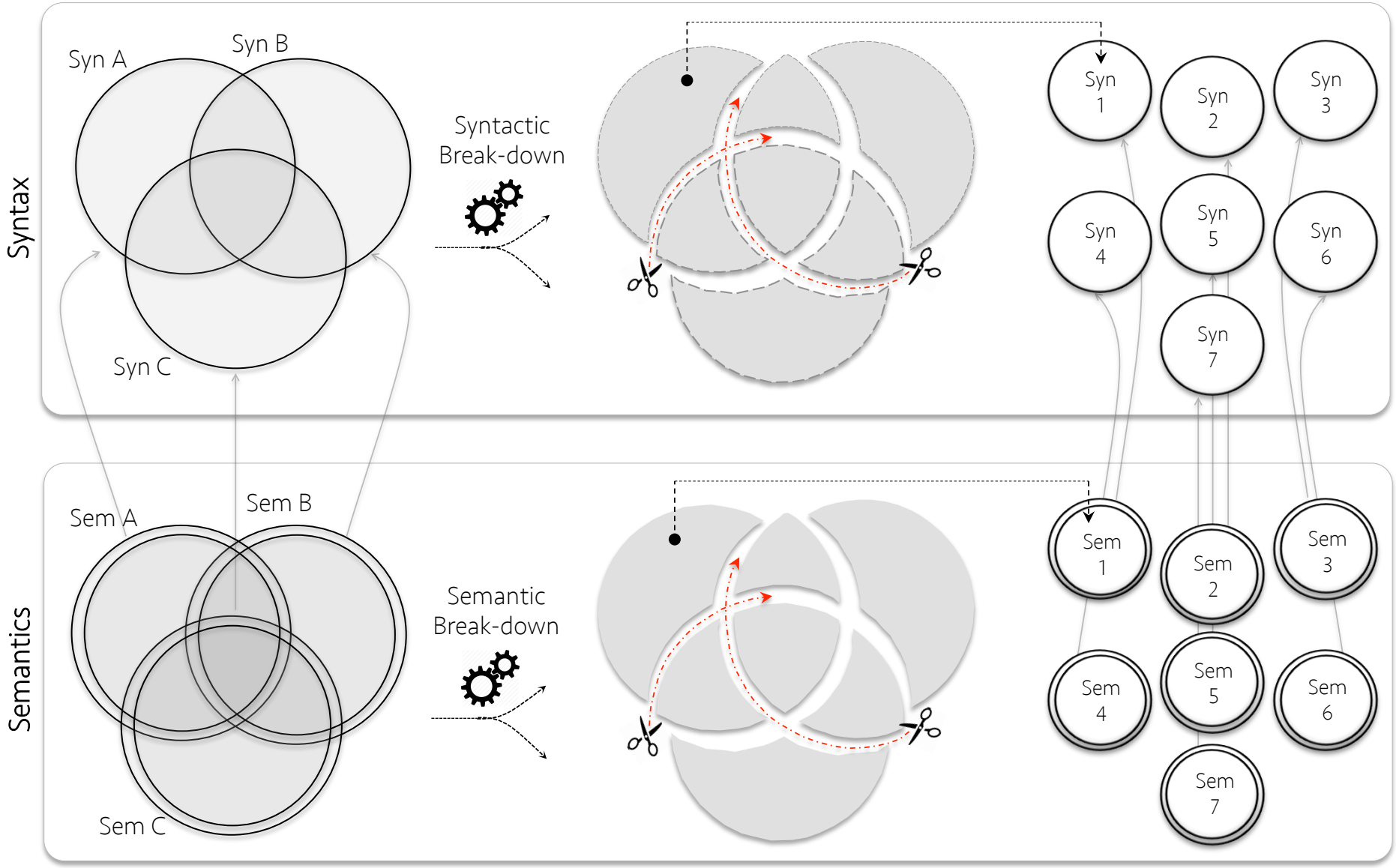

Syntactic commonalities
Semantic commonalities
Syn A
Syn B
Sem A
Sem B
MC X
MC Y
MC 0
MC Q
MC Z
DSA 1
DSA 6
DSA 2
DSA 3
DSA 4
Syn C
MC R
DSA 5
Sem C
Semantical variability


Syntactic commonalities
* State
* Constraint
* Transition
* LogoProgram
* StateMachine
* Expression
* Flowchart
* Metaclass
Semantic commonalities
- State.eval()
- Constraint.eval()
- Transition.fire()
- LogoPro.eval()
- StateMachine.eval()
- Expression.eval()
- Flowchart.eval()
- Domain-specific action

Size of Commonality (SoC)	Product Related Reusability (PRR_i)	Individualization Ratio (IR_i)	Pair-wise Relationship Ratio ($PWRR_{(i,j)}$)
			


AS
FSM
StateMachine
states *
transitions *
State
Transition
actions *
guard 0..1
Expression
Constraint
AS
Logo
LogoProgram
ins *
Primitive
kind : PrimitiveKind
params *
PrimitiveKind
Forward
Back
Left
Right
Expression
AS
Flowchart
Flowchart
arcs *
Arc
nodes *
Node
Action
Decision
actions *
guard *
Expression
Constraint
WEAVING
WEAVING
WEAVING
Sem
@Aspect(StateMachine)
execute() : void
step() : void
@Aspect(State)
do() : void
@Aspect(Transition)
fire() : void
@Aspect(Expression)
eval () : void
@Aspect(Constraint)
eval () : bool
Sem
@Aspect(LogoProgram)
execute() : void
@Aspect(Primitive)
eval() : void
@Aspect(Expression)
eval () : void
Sem
@Aspect(Arc)
fire() : void
@Aspect(Flowchart)
execute() : void
@Aspect(Action)
do() : void
@Aspect(Decision)
eval() : void
@Aspect(Expression)
eval () : void
@Aspect(Constraint)
eval () : bool


Language designers
Expressions
Constraints
State, Transition, Trigger, ...
FSM
clone
clone
Expressions
Backward, Forward, Left ...
Logo
clone
Expressions
Constraints
Node, Decision, Arc, ...
Flowchart


Syntax
Syn A
Syn B
Syn C
Syntactic
Break-down
Syn
1
Syn
2
Syn
3
Syn
4
Syn
5
Syn
6
Syn
7
Semantics
Sem A
Sem B
Sem C
Semantic
Break-down
Sem
1
Sem
2
Sem
3
Sem
4
Sem
5
Sem
6
Sem
7
First: Identifying
overlapping
Second: Breaking-
down the input set
Third: Encapsulating
language modules

	MM1	MM2	MM3
MM1	-	|MM2 ∩ MM1|	|MM3 ∩ MM1|
MM2	|MM1 ∩ MM2|	-	|MM3 ∩ MM2|
MM3	|MM1 ∩ MM3|	|MM2 ∩ MM3|	-