# The Abstract Interpreter's Control Flow
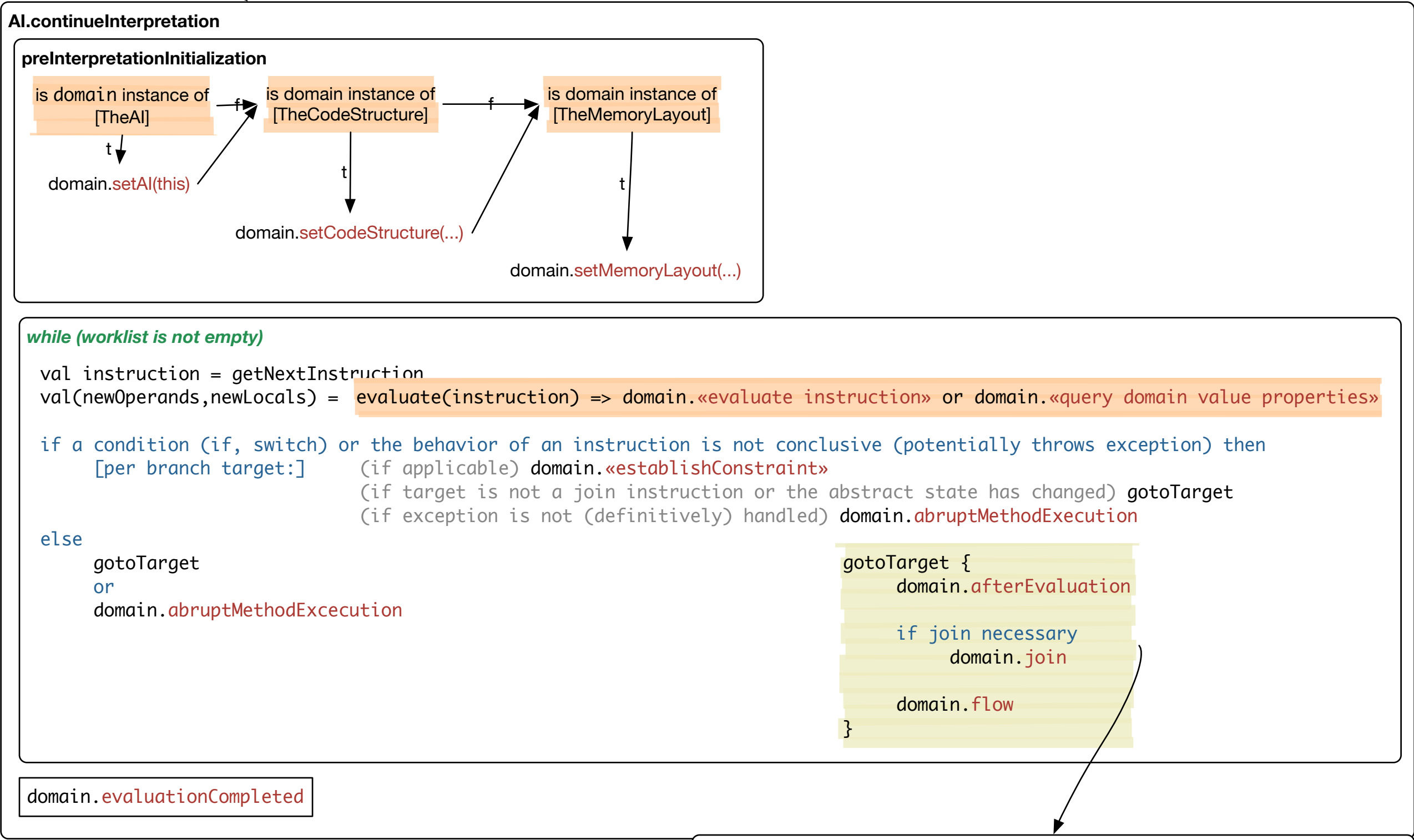
```
AI.apply(…)
```

**AI.perform(ClassFile,Method,Domain)**

initialLocals(…)    initialOperands(…)

**AI.perform(Code,Domain)(Operands,Locals)**

**AI.continueInterpretation**

**preInterpretationInitialization**

is *domain* instance of [TheAI] —f→ is domain instance of [TheCodeStructure] —f→ is domain instance of [TheMemoryLayout]

t ↓ domain.setAI(this)

t ↓ domain.setCodeStructure(…)

t ↓ domain.setMemoryLayout(…)

*while (worklist is not empty)*

```
val instruction = getNextInstruction
val(newOperands,newLocals) = evaluate(instruction) => domain.«evaluate instruction» or domain.«query domain value properties»

if a condition (if, switch) or the behavior of an instruction is not conclusive (potentially throws exception) then
    [per branch target:]     (if applicable) domain.«establishConstraint»
                             (if target is not a join instruction or the abstract state has changed) gotoTarget
                             (if exception is not (definitively) handled) domain.abruptMethodExecution
else
    gotoTarget
    or
    domain.abruptMethodExcecution
```

```
gotoTarget {
    domain.afterEvaluation

    if join necessary
        domain.join

    domain.flow
}
```

```
domain.evaluationCompleted
```

**CoreDomainFunctionality**.join

this.beforeBaseJoin {event}
[for each pair of values that need to be joined:] this.joinValues
this.afterBaseJoin {event}

this.joinPostProcessing

**CorrelationalDomainSupport**

override joinPostProcessing

**JoinStabilization**

override joinValues
override afterBaseJoin