**InterproceduralAnalysis.cpp --Class InterProceduralAnalysis<M,N,A> in soot**

A generic inter-procedural analysis which is fully context-sensitive.

This class is a base for forward and backward inter-procedural analysis classes. This inter-procedural analysis framework is fully context sensitive even in the presence of recursion and uses data flow values reaching a method to distinguish contexts.

**Data structures (Defined in interprocedural\_analysis.h)**

|  |  |
| --- | --- |
| protected  [Map](http://download.oracle.com/javase/6/docs/api/java/util/Map.html?is-external=true)<[M](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html" \o "type parameter in InterProceduralAnalysis),[List](http://download.oracle.com/javase/6/docs/api/java/util/List.html?is-external=true)<[Context](http://padhye.org/vasco/apidocs/vasco/Context.html)<[M](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[N](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[A](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html)>>> | [**contexts**](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html#contexts) |

std::unordered\_map<M, std::vector<std::reference\_wrapper<Context<M, N, A>>>> contexts;

A mapping from methods to a list of contexts for quick lookups.

|  |  |
| --- | --- |
| protected  [ContextTransitionTable](http://padhye.org/vasco/apidocs/vasco/ContextTransitionTable.html" \o "class in vasco)<[M](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[N](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[A](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html)> | [**contextTransitions**](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html#contextTransitions) |

ContextTransitionTable<M, N, A> context\_transitions;

A record of transitions from calling context and call-site to called method and called context.

|  |  |
| --- | --- |
| protected  [NavigableSet](http://download.oracle.com/javase/6/docs/api/java/util/NavigableSet.html?is-external=true" \o "class or interface in java.util)<[Context](http://padhye.org/vasco/apidocs/vasco/Context.html)<[M](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[N](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[A](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html)>> | [**worklist**](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html#worklist) |

std::vector<std::reference\_wrapper<Context<M, N, A>>> workList;

A work-list of contexts to process.

|  |  |
| --- | --- |
| protected  boolean | [**reverse**](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html#reverse) |

public:

InterProceduralAnalysis(bool reverse): reverse(reverse) { }

true if the direction of analysis is backward, or false if it is forward.

**Functions: soot vs llvm:**

|  |  |
| --- | --- |
| [Set](http://download.oracle.com/javase/6/docs/api/java/util/Set.html?is-external=true)<[CallSite](http://padhye.org/vasco/apidocs/vasco/CallSite.html" \o "class in vasco)<[M](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[N](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[A](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html)>> | [**getCallers**](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html#getCallers(vasco.Context))([Context](http://padhye.org/vasco/apidocs/vasco/Context.html)<[M](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[N](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[A](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html)> target) |

std::vector<CallSite<M, N, A>> InterProceduralAnalysis<M, N, A>::getCallers(Context<M, N, A> target) {

return context\_transitions.getCallers(target);

}

Returns the callers of a value context.

**Parameters:**

target - the value context

**Returns:**

the call-sites which transition to the value context

|  |  |
| --- | --- |
| [Context](http://padhye.org/vasco/apidocs/vasco/Context.html)<[M](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[N](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[A](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html)> | [**getContext**](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html#getContext(M,%20A))([M](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html) method, [A](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html) value) |

Retrieves a particular value context if it has been constructed.

**Parameters:**

method - the method whose value context to find

value - the data flow value at the entry (forward flow) or exit (backward flow) of the method

**Returns:**

the value context, if one is found with the given parameters, or null otherwise

std::reference\_wrapper<Context<M, N, A>> InterProceduralAnalysis<M, N, A>::getContext(M method, A value) {

if (contexts.find(method) == contexts.end()) {

return std::ref(\*(new Context<M, N, A>()));

}

if (reverse) {

for(auto e: contexts[method]) {

if(isEqual(value, e.get().getExitValue())) {

return e;

}

}

} else {

for(auto e: contexts[method]) {

if(isEqual(value, e.get().getEntryValue())) {

return e;

}

}

}

return std::ref(\*(new Context<M, N, A>()));

}

|  |  |
| --- | --- |
| [List](http://download.oracle.com/javase/6/docs/api/java/util/List.html?is-external=true)<[Context](http://padhye.org/vasco/apidocs/vasco/Context.html)<[M](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[N](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[A](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html)>> | [**getContexts**](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html#getContexts(M))([M](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html) method) |

std::vector<std::reference\_wrapper<Context<M, N, A>>> InterProceduralAnalysis<M, N, A>::getContexts(M method) {

if(contexts.find(method) == contexts.end()) {

return std::vector<std::reference\_wrapper<Context<M, N, A>>>();

}

return contexts[method];

}

Returns a list of value contexts constructed for a given method.

**Parameters:**

method - the method whose contexts to retrieve

**Returns:**

an unmodifiable list of value contexts of the given method

|  |  |
| --- | --- |
| [ContextTransitionTable](http://padhye.org/vasco/apidocs/vasco/ContextTransitionTable.html" \o "class in vasco)<[M](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[N](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[A](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html)> | [**getContextTransitionTable**](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html#getContextTransitionTable())() |

ContextTransitionTable<M, N, A> InterProceduralAnalysis<M, N, A>::getContextTransitionTable() {

return context\_transitions;

}

Returns a reference to the context transition table used by this analysis.

|  |  |
| --- | --- |
| [Set](http://download.oracle.com/javase/6/docs/api/java/util/Set.html?is-external=true)<[M](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html)> | [**getMethods**](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html#getMethods())() |

std::unordered\_set<M> InterProceduralAnalysis<M, N, A>::getMethods() {

std::unordered\_set<M> r;

for(auto e: contexts) {

r.insert(e.first);

}

return r;

}

Returns all methods for which at least one context was created.

**Returns:**

an unmodifiable set of analysed methods

|  |  |
| --- | --- |
| [Map](http://download.oracle.com/javase/6/docs/api/java/util/Map.html?is-external=true)<[M](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html" \o "type parameter in InterProceduralAnalysis),[Context](http://padhye.org/vasco/apidocs/vasco/Context.html)<[M](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[N](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[A](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html)>> | [**getTargets**](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html#getTargets(vasco.CallSite))([CallSite](http://padhye.org/vasco/apidocs/vasco/CallSite.html" \o "class in vasco)<[M](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[N](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html),[A](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html)> callSite) |

std::unordered\_map<M, Context<M, N, A>> InterProceduralAnalysis<M, N, A>::getTargets(CallSite<M, N, A> call\_site) {

return context\_transitions.getTargets(call\_site);

}

Returns the target of a call-site.

**Parameters:**

callSite - the call-site whose targets to retrieve

**Returns:**

a map of target methods to their respective contexts

std::reference\_wrapper<Context<M, N, A>> InterProceduralAnalysis<M, N, A>::getContextbyId(M method, int id) {

for(auto e: contexts[method]) {

if(id == e.get().getId()) {

return e;

}

}

}

|  |  |
| --- | --- |
| abstract  [A](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html) | [**copy**](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html#copy(A))([A](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html) src)           Returns a copy of the given data flow value. |
| abstract  [A](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html) | [**meet**](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html#meet(A,%20A))**(**[**A**](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html)**op1,**[**A**](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html)**op2)           Returns the meet of two data flow values.** |
| abstract  [A](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html) | [**topValue**](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html#topValue())**()           Returns the default data flow value (lattice top).** |
| abstract  void | [**doAnalysis**](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html#doAnalysis())**()           Performs the actual data flow analysis.** |
| abstract  [A](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html) | [boundaryValue](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html#boundaryValue(M))([M](http://padhye.org/vasco/apidocs/vasco/InterProceduralAnalysis.html) entryPoint) |

Returns the initial data flow value at the program entry points. For forward analyses this is the IN value at the ENTRY to each entry method, while for backward analyses this is the OUT value at the EXIT to each entry method.

Note that this method will be called exactly once per entry point specified by the program representation.

**Parameters:**

entryPoint - an entry point specified by the program representation

**Returns:**

the data flow value at the boundary