joern> import io.shiftleft.codepropertygraph.Cpg

| import io.shiftleft.semanticcpg.language.\_

| import io.shiftleft.codepropertygraph.cpgloading.CpgLoader

|

| val cpg = CpgLoader.load("/exports/eddie/scratch/s2291592/cpg.bin")

|

| // check node

| val nodeTypes = cpg.all.map(\_.label).groupBy(identity).view.mapValues(\_.size).toMap

| nodeTypes.foreach { case (nodeType, count) =>

| println(s"$nodeType: $count")

| }

|

| // list all method and name

| cpg.method.name.foreach(println)

|

initialising from existing storage (/exports/eddie/scratch/s2291592/cpg.bin)

NAMESPACE\_BLOCK: 3

FIELD\_IDENTIFIER: 67

BINDING: 8

TYPE: 18

TYPE\_DECL: 19

MODIFIER: 4

METHOD\_PARAMETER\_IN: 17

IDENTIFIER: 165

META\_DATA: 1

CALL: 193

METHOD: 10

RETURN: 12

FILE: 2

LITERAL: 22

LOCAL: 13

CONTROL\_STRUCTURE: 21

METHOD\_RETURN: 10

UNKNOWN: 7

BLOCK: 29

<global>

\_\_videobuf\_alloc

videobuf\_queue\_vmalloc\_init

\_\_videobuf\_mmap\_mapper

videobuf\_vmalloc\_free

videobuf\_vm\_close

\_\_videobuf\_mmap\_free

\_\_videobuf\_sync

videobuf\_to\_vmalloc

<global>

val cpg: io.shiftleft.codepropertygraph.generated.Cpg = Cpg[Graph[621 nodes]]

val nodeTypes: Map[String, Int] = HashMap(

"NAMESPACE\_BLOCK" -> 3,

"FIELD\_IDENTIFIER" -> 67,

"BINDING" -> 8,

"TYPE" -> 18,

"TYPE\_DECL" -> 19,

"MODIFIER" -> 4,

"METHOD\_PARAMETER\_IN" -> 17,

"IDENTIFIER" -> 165,

"META\_DATA" -> 1,

"CALL" -> 193,

"METHOD" -> 10,

"RETURN" -> 12,

"FILE" -> 2,

"LITERAL" -> 22,

"LOCAL" -> 13,

"CONTROL\_STRUCTURE" -> 21,

"METHOD\_RETURN" -> 10,

"UNKNOWN" -> 7,

"BLOCK" -> 29

)

joern> import io.shiftleft.codepropertygraph.Cpg

| import io.shiftleft.semanticcpg.language.\_

| import io.shiftleft.codepropertygraph.cpgloading.CpgLoader

|

| val cpg = CpgLoader.load("/exports/eddie/scratch/s2291592/cpg.bin")

|

| // check all node type and number

| val nodeTypes = cpg.all.map(\_.label).groupBy(identity).view.mapValues(\_.size).toMap

| nodeTypes.foreach { case (nodeType, count) =>

| println(s"$nodeType: $count")

| }

|

| // list all method and name

| val methodNames = cpg.method.name.l

| methodNames.foreach { methodName =>

| val cfgNodes = cpg.method.name(methodName).cfgNode.l

| println(s"method $methodName CFG node number: ${cfgNodes.size}")

| cfgNodes.foreach(println)

| }

|

initialising from existing storage (/exports/eddie/scratch/s2291592/cpg.bin)

NAMESPACE\_BLOCK: 3

FIELD\_IDENTIFIER: 67

BINDING: 8

TYPE: 18

TYPE\_DECL: 19

MODIFIER: 4

METHOD\_PARAMETER\_IN: 17

IDENTIFIER: 165

META\_DATA: 1

CALL: 193

METHOD: 10

RETURN: 12

FILE: 2

LITERAL: 22

LOCAL: 13

CONTROL\_STRUCTURE: 21

METHOD\_RETURN: 10

UNKNOWN: 7

BLOCK: 29

method <global> CFG node number: 0

method \_\_videobuf\_alloc CFG node number: 0

method videobuf\_queue\_vmalloc\_init CFG node number: 0

method \_\_videobuf\_mmap\_mapper CFG node number: 0

method videobuf\_vmalloc\_free CFG node number: 0

method videobuf\_vm\_close CFG node number: 0

method \_\_videobuf\_mmap\_free CFG node number: 0

method \_\_videobuf\_sync CFG node number: 0

method videobuf\_to\_vmalloc CFG node number: 0

method <global> CFG node number: 0

val cpg: io.shiftleft.codepropertygraph.generated.Cpg = Cpg[Graph[621 nodes]]

val nodeTypes: Map[String, Int] = HashMap(

"NAMESPACE\_BLOCK" -> 3,

"FIELD\_IDENTIFIER" -> 67,

"BINDING" -> 8,

"TYPE" -> 18,

"TYPE\_DECL" -> 19,

"MODIFIER" -> 4,

"METHOD\_PARAMETER\_IN" -> 17,

"IDENTIFIER" -> 165,

"META\_DATA" -> 1,

"CALL" -> 193,

"METHOD" -> 10,

"RETURN" -> 12,

"FILE" -> 2,

"LITERAL" -> 22,

"LOCAL" -> 13,

"CONTROL\_STRUCTURE" -> 21,

"METHOD\_RETURN" -> 10,

"UNKNOWN" -> 7,

"BLOCK" -> 29

)

val methodNames: List[String] = List(

"<global>",

"\_\_videobuf\_alloc",

"videobuf\_queue\_vmalloc\_init",

"\_\_videobuf\_mmap\_mapper",

"videobuf\_vmalloc\_free",

"videobuf\_vm\_close",

"\_\_videobuf\_mmap\_free",

"\_\_videobuf\_sync",

"videobuf\_to\_vmalloc",

"<global>"

)