



Using CDT APIs to programmatically introspect C/C++ code

Markus Schorn, Wind River Systems, Inc.

WIND RIVER

© Wind River Systems, released under EPL 1.0. All logos are TM of their respective companies.

Overview



- Different Models
 - C-Model
 - C-Index
 - Abstract Syntax Tree (AST)
 - Index-based AST
- CDT's Architecture for creating the above models
 - Scalability of model creation
- Examples for using the different Models
 - How to access the different models
- Questions and Suggestions

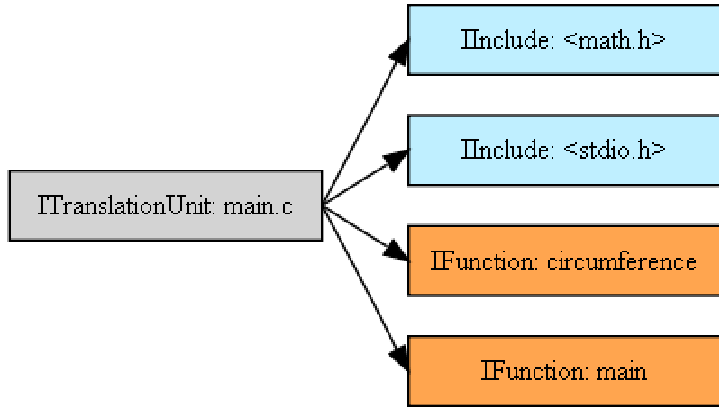
C-Model



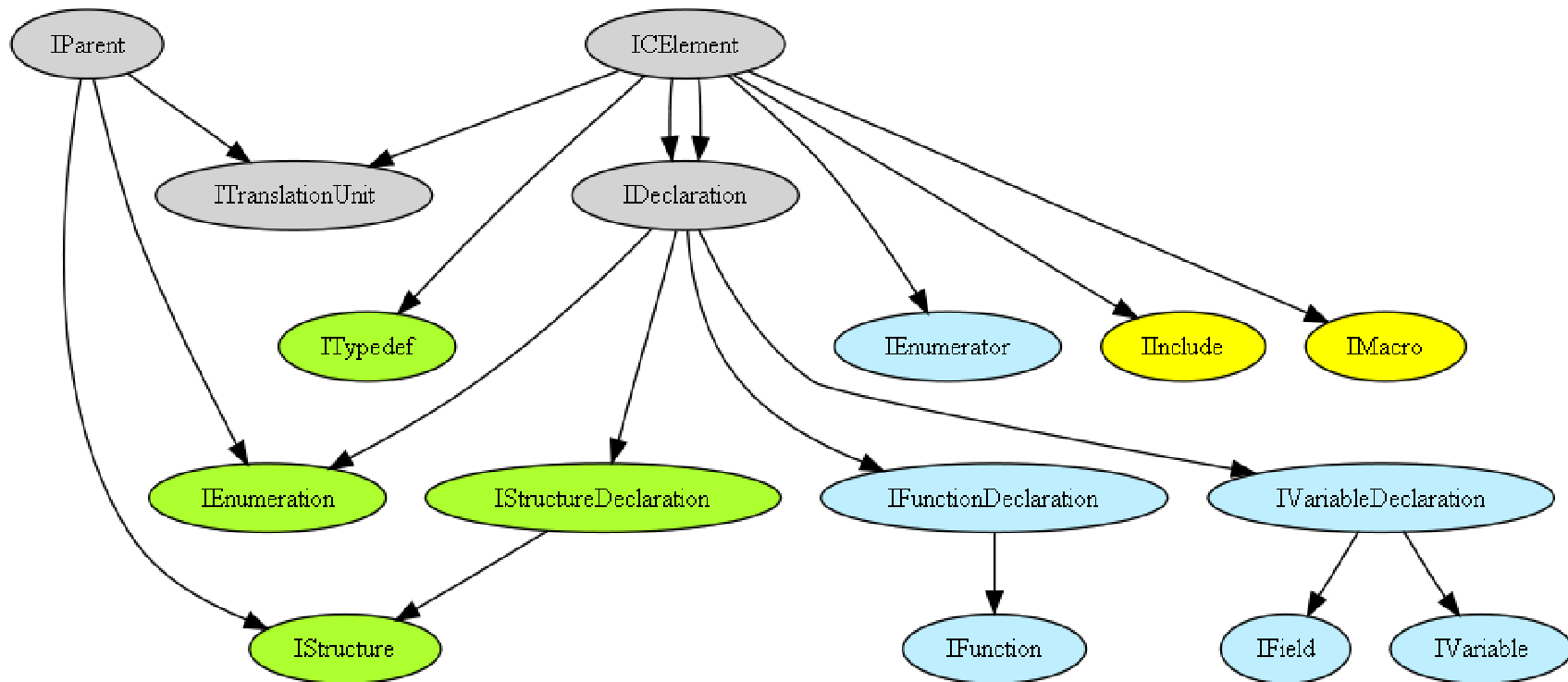
```
#include <math.h>
#include <stdio.h>

double circumference(double diam) {
    return diam * M_PI;
}

int main(int argc, char **argv) {
    double c,d;
    sscanf(argv[1], "%lf", &d);
    c= circumference(d);
    printf("c(%f)=%f", d, c);
    return 0;
}
```



ICElements



C-Model Summary



- C-Model contains information for an outline in terms of ICElements
 - **Include directives** and **macro definitions**
 - **Non-local declarations**
 - **File-location** for each ICElement
- Notification mechanism for changes to a file in the editor
 - You can register your CModelChangeListener to receive delta-information for the ICElements of an open editor.
- Limited type information
 - Types (e.g of a variable) are provided in a string representation.
- No information about references.

Index

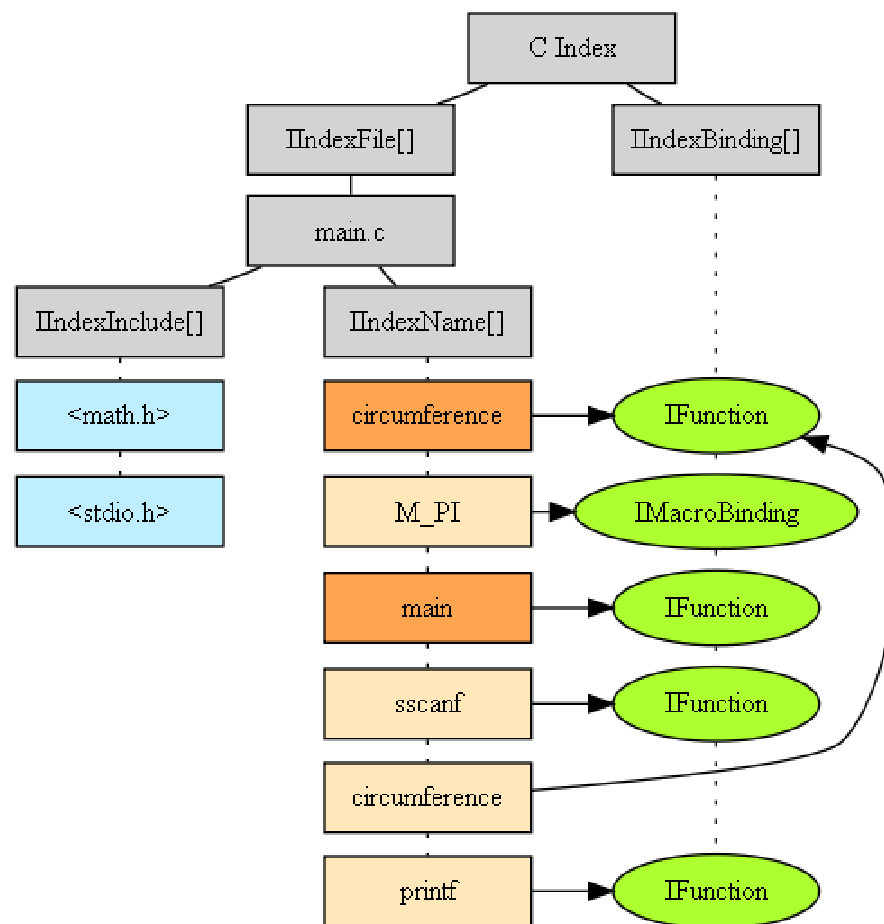
```

#include <math.h>
#include <stdio.h>

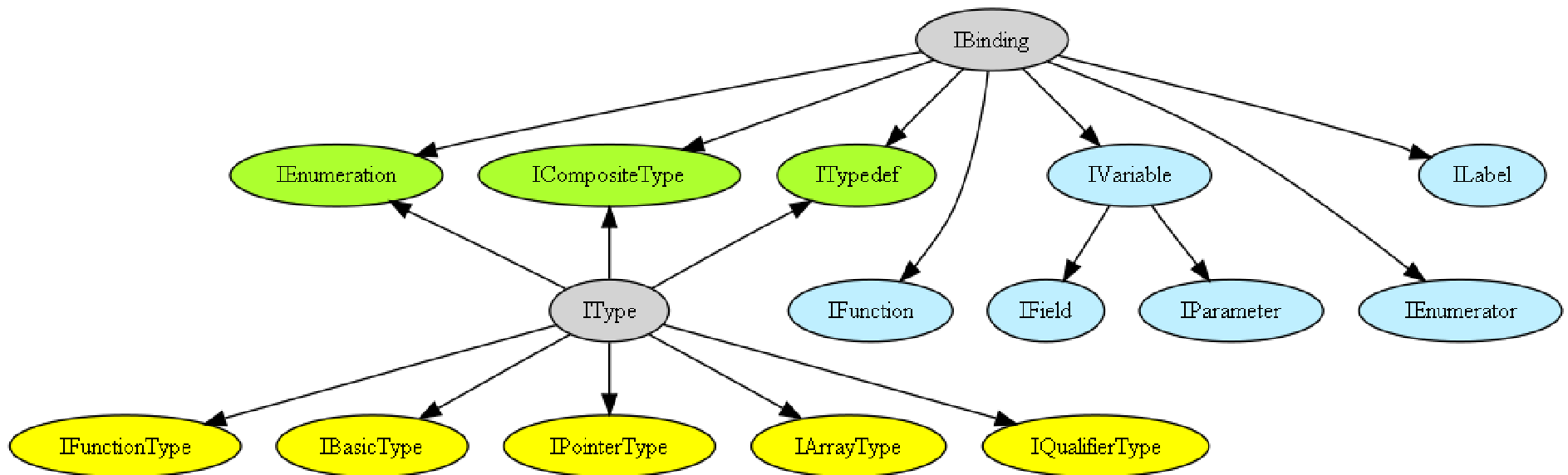
double circumference(double diam) {
    return diam * M_PI;
}

int main(int argc, char **argv) {
    double c,d;
    sscanf(argv[1], "%lf", &d);
    c= circumference(d);
    printf("c(%f)=%f", d, c);
    return 0;
}

```



Bindings and Types

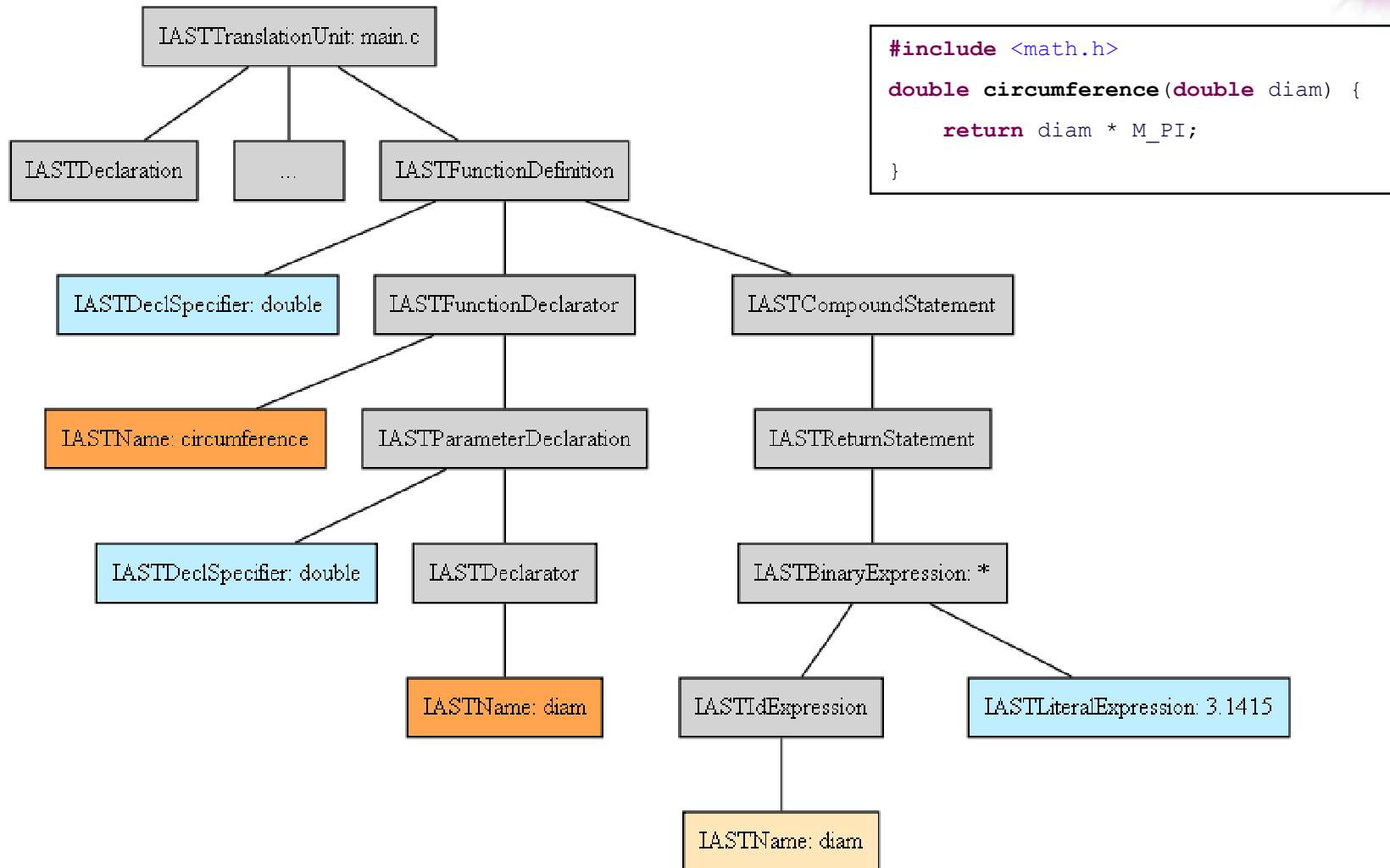


Index Summary

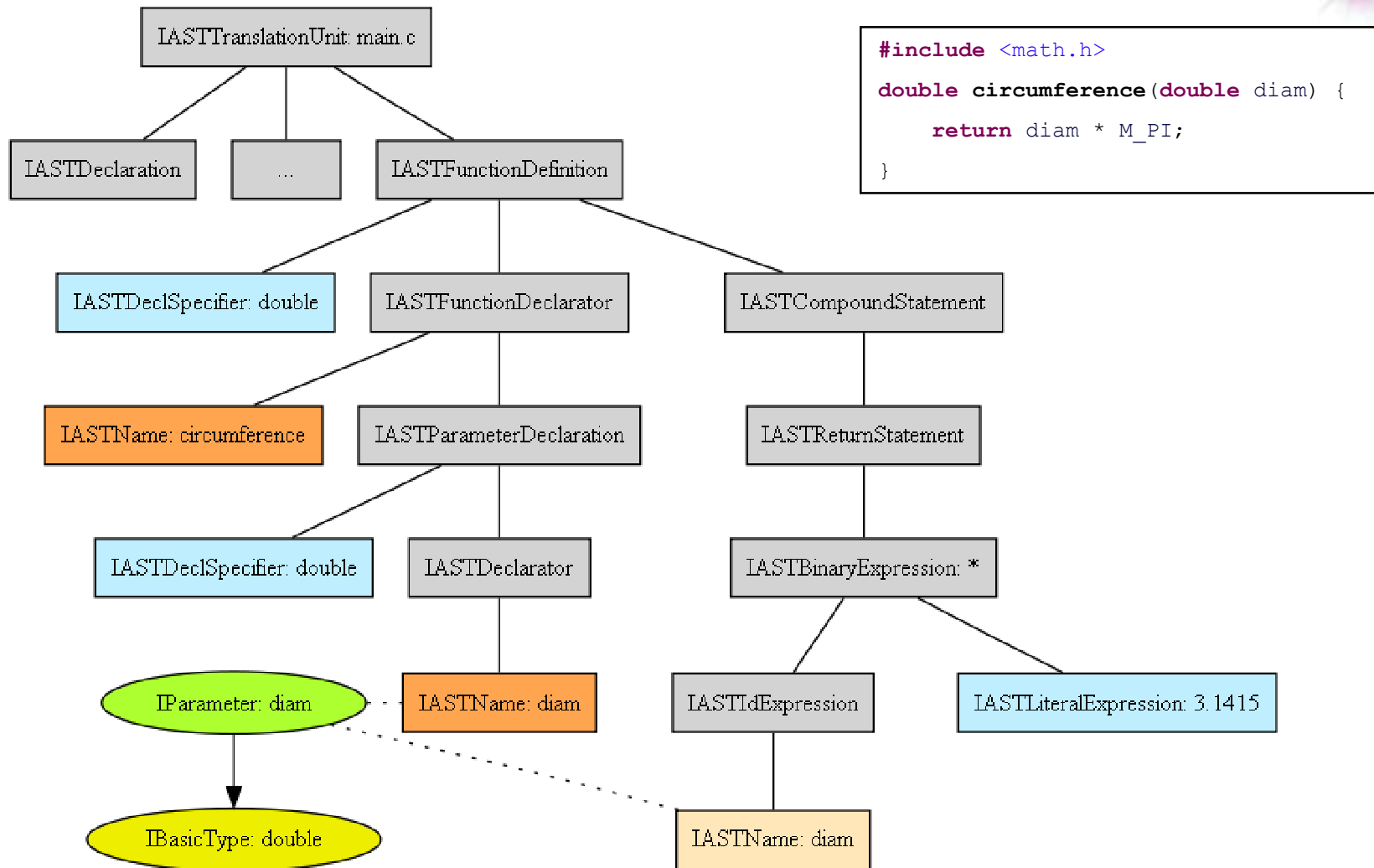


- Index contains information of global interest
 - **Include directives** and **macro definitions**
 - **Non-local declarations**
 - **References** to macros and non-local declarations
 - **File-location** for each include, macro definition, declaration and reference
 - **Binding** for each name
- Notification mechanism for changes to the model
 - You can register your `IIndexChangeListener` to get informed about changes to the index.
- Bindings and Types completely represent C/C++ entities
 - Type of a variable, return type and parameters for a function.
 - Fields of a composite type, owner of a field.

Abstract Syntax Tree



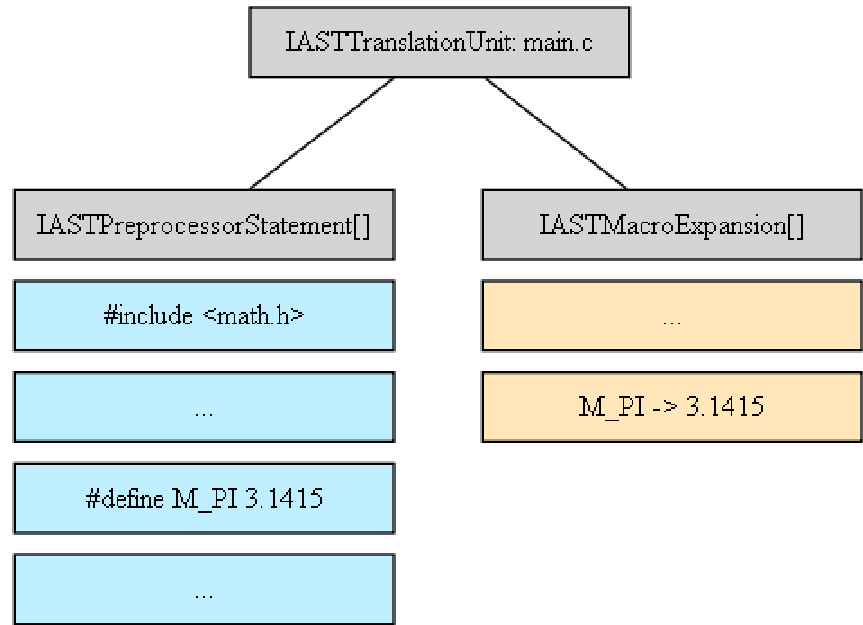
Semantics of names



Preprocessor Nodes



```
#include <math.h>
double circumference(double diam) {
    return diam * M_PI;
}
```

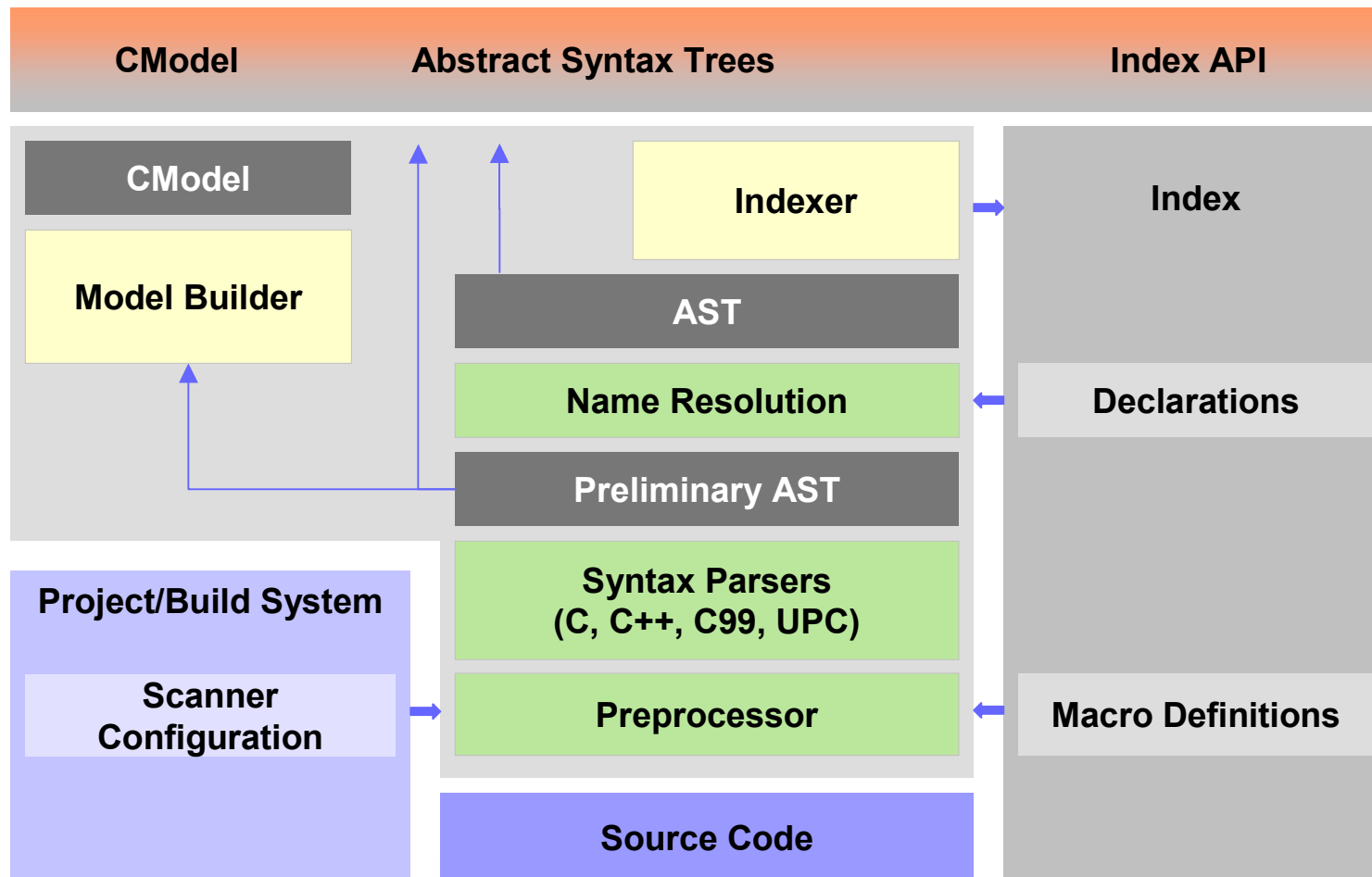


AST Summary



- AST contains every little detail about the code
 - Preprocessor directives, macro expansions and also comments.
 - References to macro-expansions.
 - Tree of nodes representing the syntax along the C/C++ grammar.
 - Includes names for all declarations and references.
 - File-location for each node with links to macro-expansions.
 - Binding for each name
- Bindings and Types completely represent C/C++ entities
 - Type of a variable, return type and parameters for a function.
 - Fields of a composite type, owner of a field.
- Scopes (important for local bindings, only)

CDT's Parsing Architecture



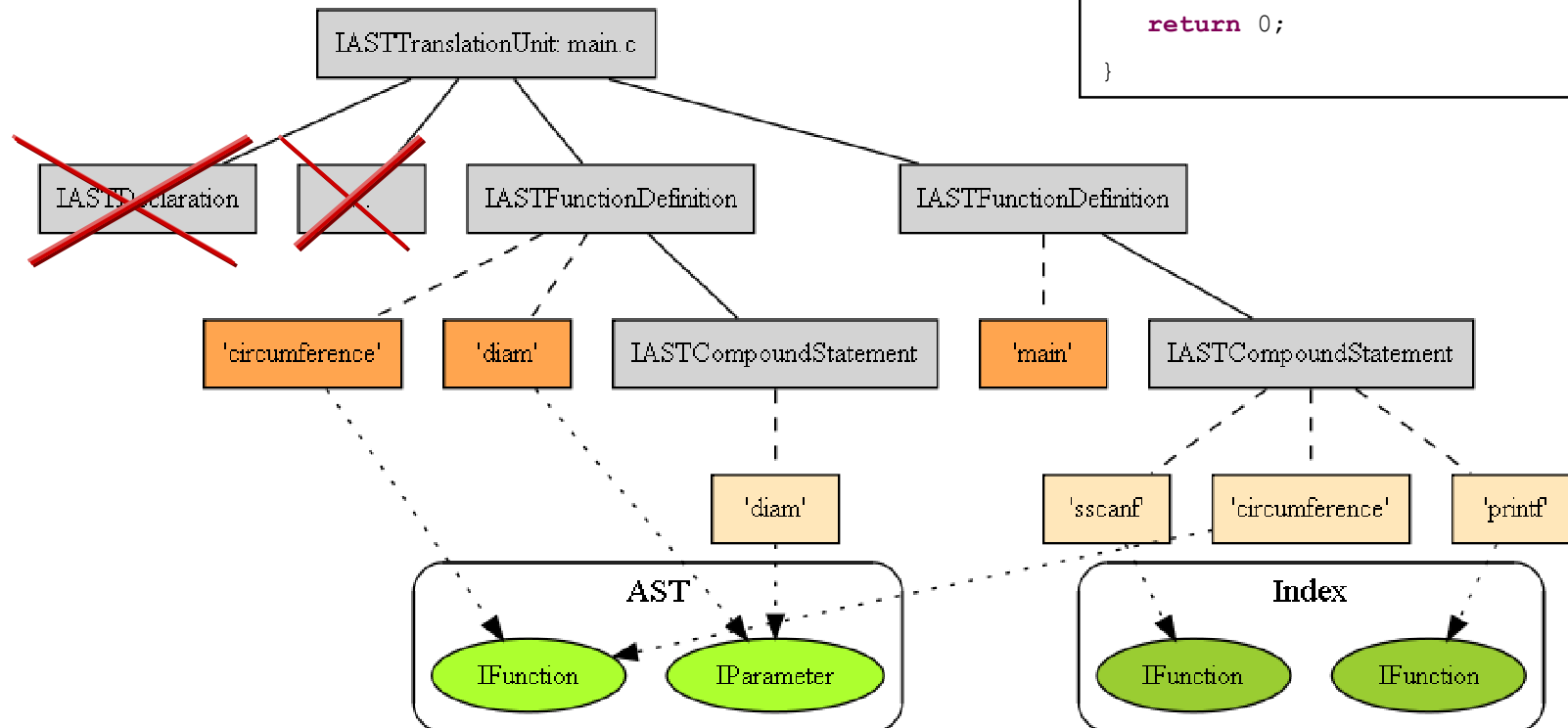


Index-based AST

```
// main.c
#include <math.h>
#include <stdio.h>

double circumference(double diam) {
    return diam * M_PI;
}

int main(int argc, char **argv) {
    double c,d;
    sscanf(argv[1], "%lf", &d);
    c= circumference(d);
    printf("c(%f)=%f", d, c);
    return 0;
}
```



Access to C-Model and C-Index



- **C-Model:** ITranslationUnit for a workspace file

```
IPath path= new Path("project/folder/file.c");
IFile file= ResourcesPlugin.getWorkspace().getRoot().getFile(path);
// Create translation unit for file
ITranslationUnit tu= (ITranslationUnit) CoreModel.getDefault().create(file);
```

- **C-Model:** ITranslationUnit for file in the editor

```
IEditorPart e= PlatformUI.getWorkbench().getActiveWorkbenchWindow().getActivePage().getActiveEditor();
// Access translation unit of the editor.
ITranslationUnit tu= (ITranslationUnit) CDTUITools.getEditorInputCElement(editor.getEditorInput());
```

- **C-Index:** IIndex for one or more projects

```
// one project
ICProject project= CoreModel.getDefault().getCModel().getCProject("project");
IIndex index= CCorePlugin.getIndexManager().getIndex(project);

// all projects
ICProject[] allProjects= CoreModel.getDefault().getCModel().getCProjects();
index= CCorePlugin.getIndexManager().getIndex(allProjects);
```

Creating AST



- **Complete AST:** IASTTranslationUnit for a workspace file

```
ITranslationUnit tu= ...; // see previous slide
IASTTranslationUnit ast= tu.getAST(); // 'getAST()' is miss-leading, it actually creates the AST
```

- **Index-based AST:** IASTTranslationUnit for a workspace file

```
IIndex index= ...; // see previous slide
ITranslationUnit tu= ...; // see previous slide

index.acquireReadLock(); // we need a read-lock on the index
try {
    ast= tu.getAST(index, ITranslationUnit.AST_SKIP_INDEXED_HEADERS);
} finally {
    index.releaseReadLock();
    ast= null; // don't use the ast after releasing the read-lock
}
```

- **AST:** IASTTranslationUnit for a CodeReader (for experts, only)

```
ILanguage lang= GPPLanguage.getDefault();
IASTTranslationUnit ast= lang.getASTTranslationUnit(reader, scannerInfo, readerFactory, index, log);
```


Accessing Shared AST



- **Index-based AST:** IASTTranslationUnit for file in the editor

```
IEditorPart e= PlatformUI.getWorkbench().getActiveWorkbenchWindow().getActivePage().getActiveEditor();
ITranslationUnit tu= (ITranslationUnit) CDTUITools.getEditorInputCElement(editor.getEditorInput());

Job job= new SharedASTJob("Job Name", tu) {
    @Override
    public IStatus runOnAST(ILanguage lang, IASTTranslationUnit ast) throws CoreException {
        // index is locked for you, use the ast
        // ...
    }
};
job.schedule();
```

Example #1: References in Index



```
void outputReferences(String functionName) throws CoreException, InterruptedException {
    // Access index
    IProject[] allProjects= CoreModel.getDefault().getCMModel().getCProjects();
    IIndex index= CCorePlugin.getIndexManager().getIndex(allProjects);

    index.acquireReadLock(); // we need a read-lock on the index
    try {
        // find bindings for name
        IIndexBinding[] bindings= index.findBindings(functionName.toCharArray(),
            IndexFilter.ALL_DECLARED, new NullProgressMonitor());

        // find references for each binding
        for (IIndexBinding b : bindings) {
            if (b instanceof IFunction) {
                outputReferences(index, b);
            }
        }
    } finally {
        index.releaseReadLock();
    }
}

void outputReferences(IIndex index, IBinding b) throws CoreException{
    IIndexName[] names= index.findReferences(b);
    for (IIndexName n : names) {
        outputReference(index, n);
    }
}

void outputReference(IIndex index, IIndexName n) throws CoreException {
    IASTFileLocation fileLoc= n.getFileLocation();
    System.out.println(fileLoc.getFileName() + " at offset " + fileLoc.getNodeOffset());
}
```

Example #2: Enclosing function in Index



```
// See previous slide
void outputReferences(String functionName) throws CoreException, InterruptedException;

// See previous slide
void outputReferences(IIndex index, IBinding b) throws CoreException;

void outputReference(IIndex index, IIndexName n) throws CoreException {
    System.out.print(fileLoc.getFileName() + " at offset " + fileLoc.getNodeOffset());
    // Output name of enclosing function
    IIndexName within= n.getEnclosingDefinition();
    if (within != null) {
        IBinding enclosing= index.findBinding(within);
        if (enclosing instanceof IFunction) {
            System.out.print(" within " + enclosing.getName());
        }
    }
    System.out.println();
}
```

Example #3: Get selected binding using AST



```
void getSelectedBinding() {
    // find active editor
    IEditorPart e= PlatformUI.getWorkbench().getActiveWorkbenchWindow().getActivePage().getActiveEditor();
    if (!(e instanceof ITextEditor))
        return;

    // cursor position and translation unit
    final ITextSelection sel= (ITextSelection) ((ITextEditor) e).getSelectionProvider().getSelection();
    ITranslationUnit tu= (ITranslationUnit) CDTUITools.getEditorInputCElement(e.getEditorInput());

    // access shared ast
    Job job= new SharedASTJob("Get selected binding", tu) {
        @Override
        public IStatus runOnAST(ILanguage lang, IASTTranslationUnit ast) throws CoreException {
            IASTNodeSelector nodeSelector= ast.getNodeSelector(null);
            IASTName name= nodeSelector.findEnclosingName(sel.getOffset(), sel.getLength());
            if (name != null) {
                IBinding b= name.resolveBinding();

                // see example #1
                outputReferences(ast.getIndex(), b);
            }
            return Status.OK_STATUS;
        }
    };
    job.schedule();
}
```

Example #4: Find bogus assignments in AST



```
void findBogusAssignments(IFile file) throws CoreException, InterruptedException {
    // create translation unit and access index
    ITranslationUnit tu= (ITranslationUnit) CoreModel.getDefault().create(file);
    IIndex index= CCorePlugin.getIndexManager().getIndex(tu.getCProject());

    // lock the index for read access
    index.acquireReadLock();
    try {
        // create index based ast
        IASTTranslationUnit ast= tu.getAST(index, ITranslationUnit.AST_SKIP_INDEXED_HEADERS);
        // traverse the ast using the visitor pattern.
        ast.accept(new CheckCodeVisitor());
    } finally {
        index.releaseReadLock();
    }
}
```

Example #4 ...continued



```
class CheckCodeVisitor extends ASTVisitor {
    CheckCodeVisitor() {
        shouldVisitExpressions= true;
    }

    public int visit(IASTExpression expression) {
        if (isAssignmentExpression(expression) && isUsedAsCondition(expression)) {
            System.out.println("warning ..." + expression.getFileLocation());
        }
        return PROCESS_CONTINUE;
    }

    private boolean isAssignmentExpression(IASTExpression e) {
        if (e instanceof IASTBinaryExpression) {
            IASTBinaryExpression binExpr= (IASTBinaryExpression) e;
            return binExpr.getOperator() == IASTBinaryExpression.op_assign;
        }
        return false;
    }

    private boolean isUsedAsCondition(IASTExpression expression) {
        ASTNodeProperty prop = expression.getPropertyInParent();
        if (prop == IASTForStatement.CONDITION || prop == IASTIfStatement.CONDITION)
            return true;
        return false;
    }
}
```



Questions or Suggestions?