Coccinelle for Rust
https://gitlab.inria.fr/coccinelle/coccinelleforrust.git

Julia Lawall, Tathagata Roy
November 15, 2023
- Perform repetitive transformations at a large scale.
Goals

- Perform repetitive transformations at a large scale.
  - Rust is 1.6 MLOC.
  - The Linux kernel is 23 MLOC.
Goals

- Perform repetitive transformations at a large scale.
  - Rust is 1.6 MLOC.
  - The Linux kernel is 23 MLOC.
  - Collateral evolutions: a change in an API requires changes in all clients.
Goals

- Perform repetitive transformations at a large scale.
  - Rust is 1.6 MLOC.
  - The Linux kernel is 23 MLOC.
  - Collateral evolutions: a change in an API requires changes in all clients.

- Provide a transformation language that builds on developer expertise.
Goals

- Perform repetitive transformations at a large scale.
  - Rust is 1.6 MLOC.
  - The Linux kernel is 23 MLOC.
  - Collateral evolutions: a change in an API requires changes in all clients.

- Provide a transformation language that builds on developer expertise.

- Changes + developer familiarity = (semantic) patches
commit d822b97a27e50f5a091d2918f6ff0f82d2827f5
Author: Kyle Matsuda <kyle.yoshio.matsuda@gmail.com>
Date: Mon Feb 6 17:48:12 2023 -0700

change usages of type_of to bound_type_of

diff --git a/compiler/rustc_borrowck/src/diagnostics/conflict_errors.rs b/compiler/.../conflict_errors.rs
@@ -2592,4 +2592,4 @@ fn annotate_argument_and_return_for_borrow(
     ) else {
-        let ty = self.infcx.tcx.type_of(self.mir_def_id());
+        let ty = self.infcx.tcx.bound_type_of(self.mir_def_id()).subst_identity();
            match ty.kind() {
-            ty::FnDef(_, _) | ty::FnPtr(_) => self.annotate_fn_sig(
+            ty::FnDef(_, _) | ty::FnPtr(_) => self.annotate_fn_sig(

diff --git a/compiler/rustc_borrowck/src/diagnostics/mod.rs b/compiler/.../mod.rs
@@ -1185,4 +1185,4 @@ fn explaincaptures(
            matches!(tcx.def_kind(parent_did), rustc_hir::def::DefKind::Impl { .. })
                .then_some(parent_did)
-            .and_then(|did| match tcx.type_of(did).kind() {
+            .and_then(|did| match tcx.bound_type_of(did).subst_identity().kind() {
                 ty::Adt(def, ..) => Some(def.did()),
...

136 files changed, 385 insertions(+), 262 deletions(-)
An example change (Rust repository)

commit d822b97a27e50f5a091d2918f6ff0ffd2d2827f5
Author: Kyle Matsuda <kyle.yoshio.matsuda@gmail.com>
Date: Mon Feb 6 17:48:12 2023 -0700

change usages of type_of to bound_type_of

diff --git a/compiler/rustc_borrowck/src/diagnostics/conflict_errors.rs b/compiler/.../conflict_errors.rs
@@ -2592,4 +2592,4 @@ fn annotate_argument_and_return_for_borrow()
 } else {
 -   let ty = self.infcx.tcx.type_of(self.mir_def_id());
+   let ty = self.infcx.tcx.bound_type_of(self.mir_def_id()).subst_identity();
     match ty.kind() {
       ty::FnDef(_, _) | ty::FnPtr(_) => self.annotate_fn_sig(

diff --git a/compiler/rustc_borrowck/src/diagnostics/mod.rs b/compiler/.../mod.rs
@@ -1185,4 +1185,4 @@ fn explain_captures(
       matches!(tcx.def_kind(parent_did), rustc_hir::def::DefKind::Impl { .. })
         .then_some(parent_did)
 -          .and_then(|did| match tcx.type_of(did).kind() {
+          .and_then(|did| match tcx.bound_type_of(did).subst_identity().kind() {
              ty::Adt(def, ..) => Some(def.did()),
...
Creating a semantic patch: Step 1: remove irrelevant code

- let ty = self.infcx.tcx.type_of(self.mir_def_id())
+ let ty = self.infcx.tcx.bound_type_of(self.mir_def_id()).subst_identity()
Creating a semantic patch: Step 2: pick a typical example

@@

@@

- self.infcx.tcx.type_of(self.mir_def_id())
+ self.infcx.tcx.bound_type_of(self.mir_def_id()).subst_identity()
Creating a semantic patch: Step 3: abstract over subterms using metavariables

```plaintext
@@
expression tcx, arg;
@@

- tcx.type_of(arg)
+ tcx.bound_type_of(arg).subst_identity()
```
expression tcx, arg;

- tcx.type_of(arg)
+ tcx.bound_type_of(arg).subst_identity()
let (shim_size, shim_align, _kind) = ecx.get_alloc_info(alloc_id);
let def_ty = ecx.tcx.bound_type_of(def_id).subst_identity();
let extern_decl_layout =
    ecx.tcx.layout_of(ty::ParamEnv::empty().and(ecx.tcx.type_of(def_id))).unwrap();
+ ecx.tcx.layout_of(ty::ParamEnv::empty().and(def_ty)).unwrap();
if extern_decl_layout.size != shim_size || extern_decl_layout.align.abi != shim_align {
    throw_unsup_format!(
        "\"extern\" static `{name}` from crate `{krate}` has been declared \n    "
}
let (shim_size, shim_align, _kind) = ecx.get_alloc_info(alloc_id);
+
let def_ty = ecx.tcx.bound_type_of(def_id).subst_identity();
-
let extern_decl_layout =
 +
    ecx.tcx.layout_of(ty::ParamEnv::empty().and(ecx.tcx.type_of(def_id))).unwrap();
-
+    ecx.tcx.layout_of(ty::ParamEnv::empty().and(def_ty)).unwrap();

if extern_decl_layout.size != shim_size || extern_decl_layout.align.abi != shim_align {
    throw_unsup_format!(
        "\`extern` static \`{name}\` from crate \`{krate}\` has been declared \n    

The developer has created a new name to avoid a long line.

- Could address it manually.

- Could create a rule for the special case of nested function call contexts (probably not worth it for one case).
An alternate semantic patch

```
@@
expression tcx, arg;
@@

tcx.
-    type_of(arg)
+    bound_type_of(arg).subst_identity()
```

Putting tcx in the context ensures any comments will be preserved.
Specifying the type of `tcx` protects against changing other uses of `type_of`.
Some Coccinelle internals

**Input:** Parsing provided by Rust Analyzer.

- Used both for Rust code and for semantic patch code.
- Will provide type inference, when needed (currently, loses concurrency).
Some Coccinelle internals

**Input:** Parsing provided by Rust Analyzer.

- Used both for Rust code and for semantic patch code.
- Will provide type inference, when needed (currently, loses concurrency).

**Output:** Pretty printing provided by `rustfmt`.

- To avoid problems with code not originally formatted with `rustfmt` (or formatted with a different version), the `rustfmt`ed changes are dropped back into the original code.
- Preserves comments and whitespace in the unchanged part of the code.
Some Coccinelle internals

In the middle:

- Wrap Rust code and semantic patch code, eg to indicate metavariables.
- Match semantic patch code against Rust code, to collect change sites and metavariable bindings.
- On a successful match, apply the changes, instantiated according to the metavariable bindings, reparse, and repeat with the next rule.
A case study

**Software:** stratisd

- [https://github.com/stratis-storage/stratisd](https://github.com/stratis-storage/stratisd)
- Easy to use local storage management for Linux.
- Over 2000 commits since 2016, and over 10K lines of Rust code.

**Commit selection:**

- Patchparse: [https://gitlab.inria.fr/lawall/patchparse4](https://gitlab.inria.fr/lawall/patchparse4)
- Collect change patterns that occur at least 40 times.
- 13 commits selected, affecting 10-94 files, and up to 3000 $+/-$ lines.
Some successes

Commits:

- 39b925b0: Remove EngineError alias
- c3918972: Replace EngineResult usage with StratisResult

Semantic patch:

```
@type@
@@
- EngineError
+ StratisError
```

```
@type@
@@
- EngineResult
+ StratisResult
```
Some successes

Commits:

- 39b925b0: Remove EngineError alias
- c3918972: Replace EngineResult usage with StratisResult

Semantic patch:

@type@
@@
- EngineError
+ StratisError

Results:

- Typical changes: use, method signatures, method calls.
  - Not completely following the Rust Analyzer AST.
- Benefits from recent improvements in pretty printing.
Some successes

**fe7df6a9**: Remove unnecessary pub modifier on stratisd tests

**Semantic patch:**

```plaintext
@@
identifier f;
expression e;
@@
#[test]
- pub
  fn f() { e; }
```

**Results:**

- 69 changes across 9 files.
- 1 case has an additional attribute and thus is omitted.
A partial success

9c60ad44: Remove ErrorEnum and add error chaining

```c
expression return_message, e1;
@
return_message.append3(e1,
- msg_code_ok(), msg_string_ok(),
+ DbusErrorEnum::OK as u16, OK_STRING.to_string(),
)
@

- DbusErrorEnum::INTERNAL_ERROR
+ DbusErrorEnum::ERROR
```

Results:
- Covers 209/417 changes. Omits uses and some less common patterns.
- Trailing commas lead to a lot of rule duplication.
- Treatment of Error too simplistic, leading to false positives.
Another partial success

d4ac5d89: Switch from trait objects to type parameters and associated types

- Covers 111/418 changes.
- Trailing commas issues. Borrowing issues.
- New feature: ... for parameter lists and for method bodies.
  - For method bodies, matches both simple expressions and block expressions.
Some failures

Commits:

- aeed4b7c: Use inline format arguments
- ea33caf4: Conform to snake_case naming style
Some failures

Commits:

- aeed4b7c: Use inline format arguments
- ea33caf4: Conform to snake_case naming style

Issues:

- Require changes inside identifier names and strings.
- Such changes require scripting, as found in Coccinelle for C.
Some failures

2569545c: Add anonymous lifetime parameters.

Semantic patch extract:

```plaintext
@type@
lifetime l1,l2;
@@
(
App <l1,l2>
| App
+ <',','
)
```

Disjunctions on types not currently supported.
Some failures

f00fb860: Allow disabling actions when stratisd detects unresolvable failures

Semantic patch extract:

```c
@@
identifier mthd;
@@
- log_action(!
+ handle_action(!
    pool.mthd(
        ...
    )
+   ,dbus_context, pool_path.get_name()
    )
```

Issues:

- This covers a few changes, but the commit has more variety.
- New feature: ... for argument lists.
- Future feature: ... to connect the definitions of pool_path to the call site.
Discussion

- Rust projects of interest?
- Transformations of interest?
Conclusion

- Pattern-based transformation language.
  - Changes can be expressed in all parts of the code: expressions, signatures, lifetimes, etc.
  - Changes can be sensitive to expression types.

- Works well for frequent atomic changes.
  - Recent updates to improve pretty printing, handling of macros, genericity (…), etc.

- Future work: … for control-flow paths, nesting.
  - Connect variable definitions to uses.
  - Connect method definitions to the containing type implementation.

https://gitlab.inria.fr/coccinelle/coccinelleforrust.git
https://rust-for-linux.com/coccinelle-for-rust