rust-sword与Simple DB系统实践

汇报人：潘中颢

南京大学软件学院

Email: zhonghaopan@smail.nju.edu.cn

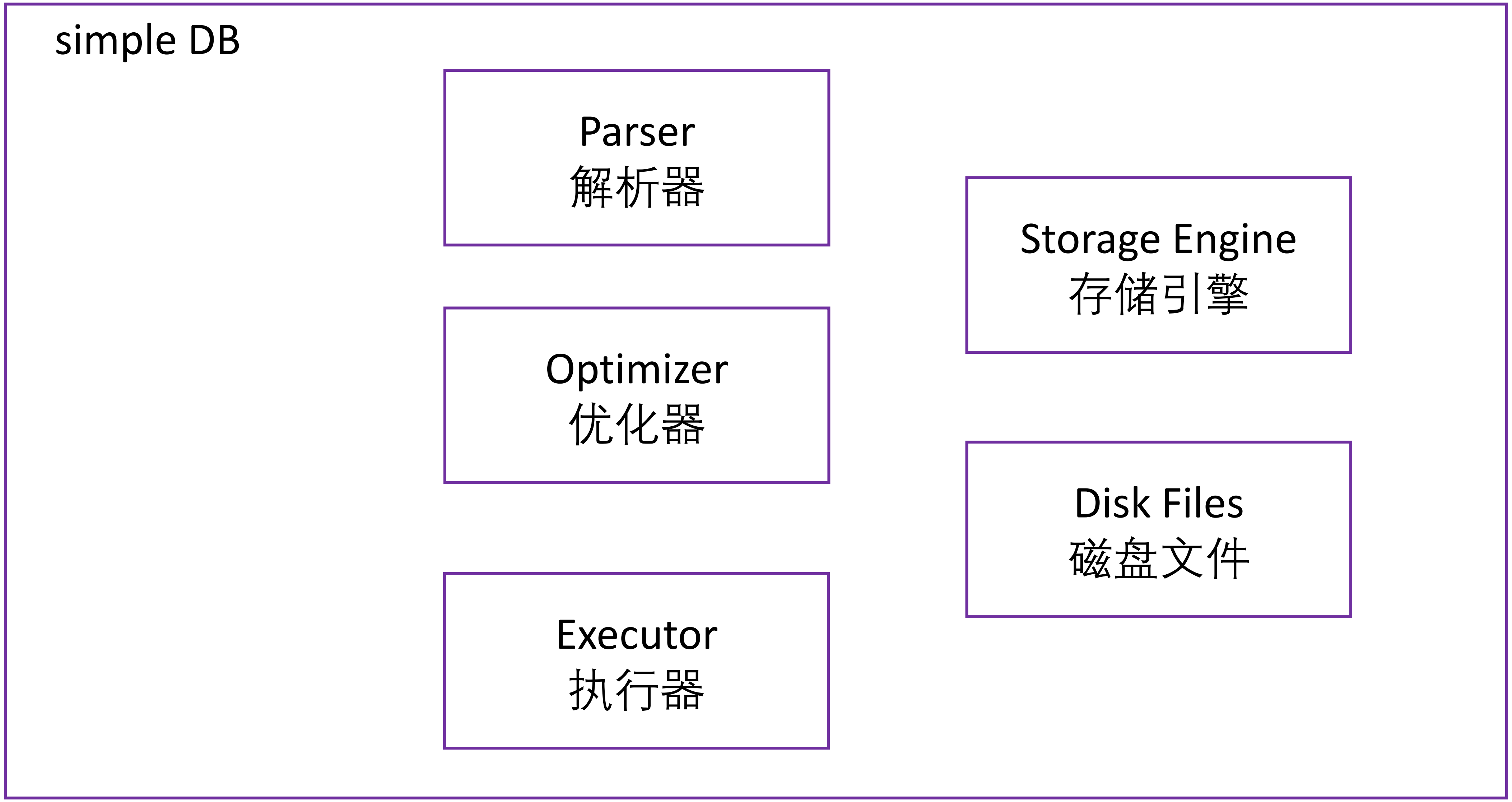
rust-sword vs Simple DB

如何实现Parser?

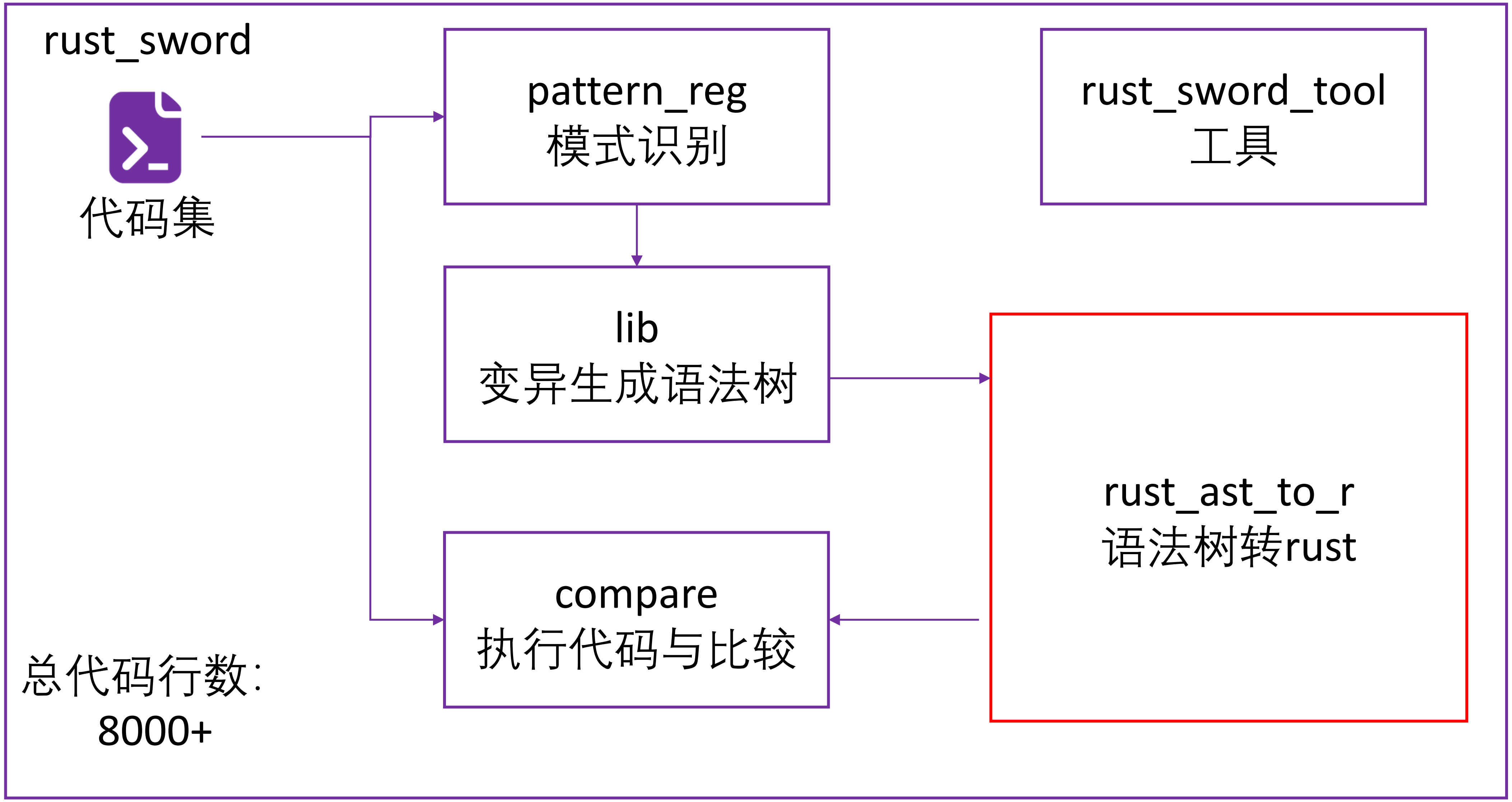
rust-sword： 代码->语法树->数据信息

simple DB example： 代码->规则解释->数据信息

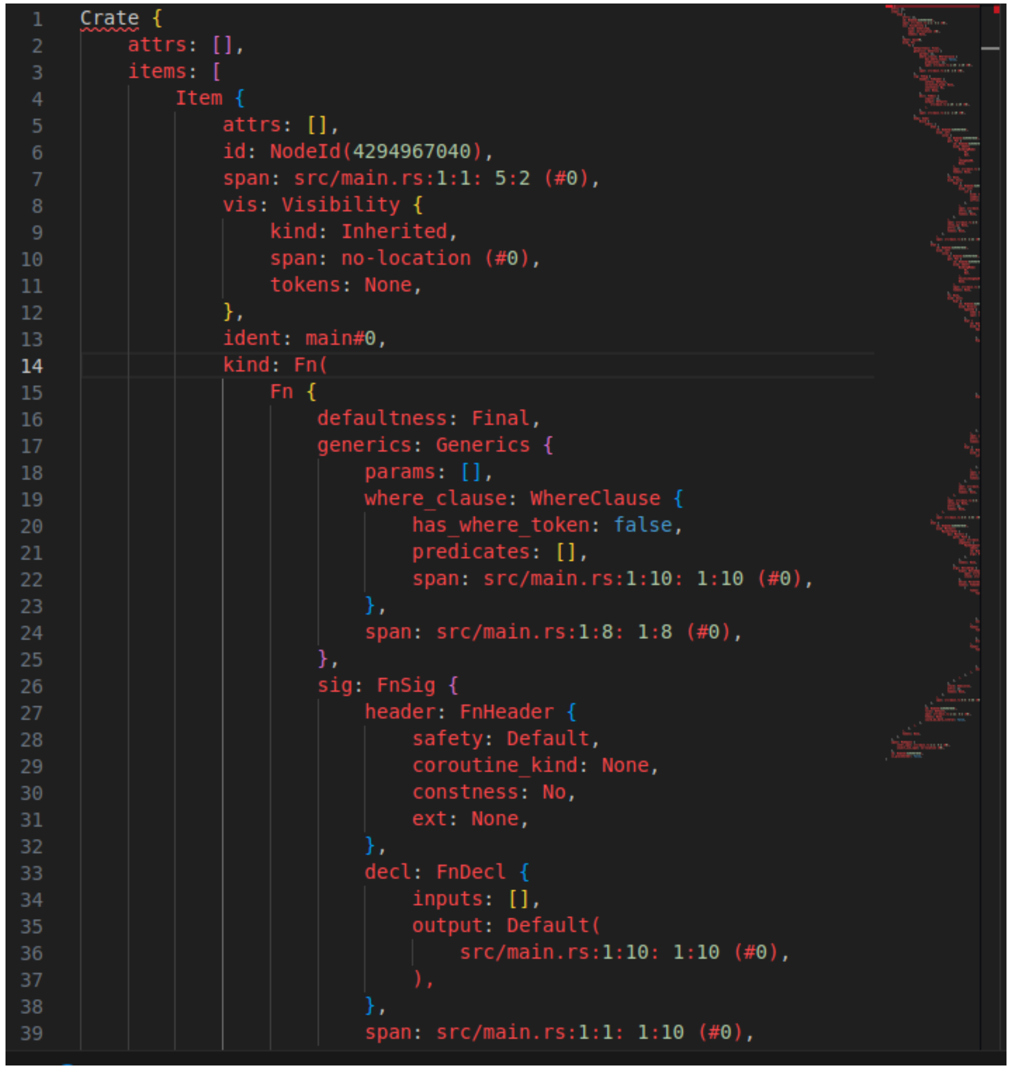
Simple DB: 系统架构



rust-sword：系统架构



rust_ast_to_r：语法树转rust



```
Crate {
    attrs: [],
    items: [
        Item {
            attrs: [],
            id: NodeId(4294967040),
            span: src/main.rs:1:1: 5:2 (#0),
            vis: Visibility {
                kind: Inherited,
                span: no-location (#0),
                tokens: None,
            },
            ident: main#0,
            kind: Fn(
                Fn {
                    defaultness: Final,
                    generics: Generics {
                        params: [],
                        where_clause: WhereClause {
                            has_where_token: false,
                            predicates: [],
                            span: src/main.rs:1:10: 1:10 (#0),
                        },
                        span: src/main.rs:1:8: 1:8 (#0),
                    },
                    sig: FnSig {
                        header: FnHeader {
                            safety: Default,
                            coroutine_kind: None,
                            constness: No,
                            ext: None,
                        },
                        decl: FnDecl {
                            inputs: [],
                            output: Default(
                                src/main.rs:1:10: 1:10 (#0),
                            ),
                        },
                        span: src/main.rs:1:1: 1:10 (#0),
```

从语法树转回rust代码：

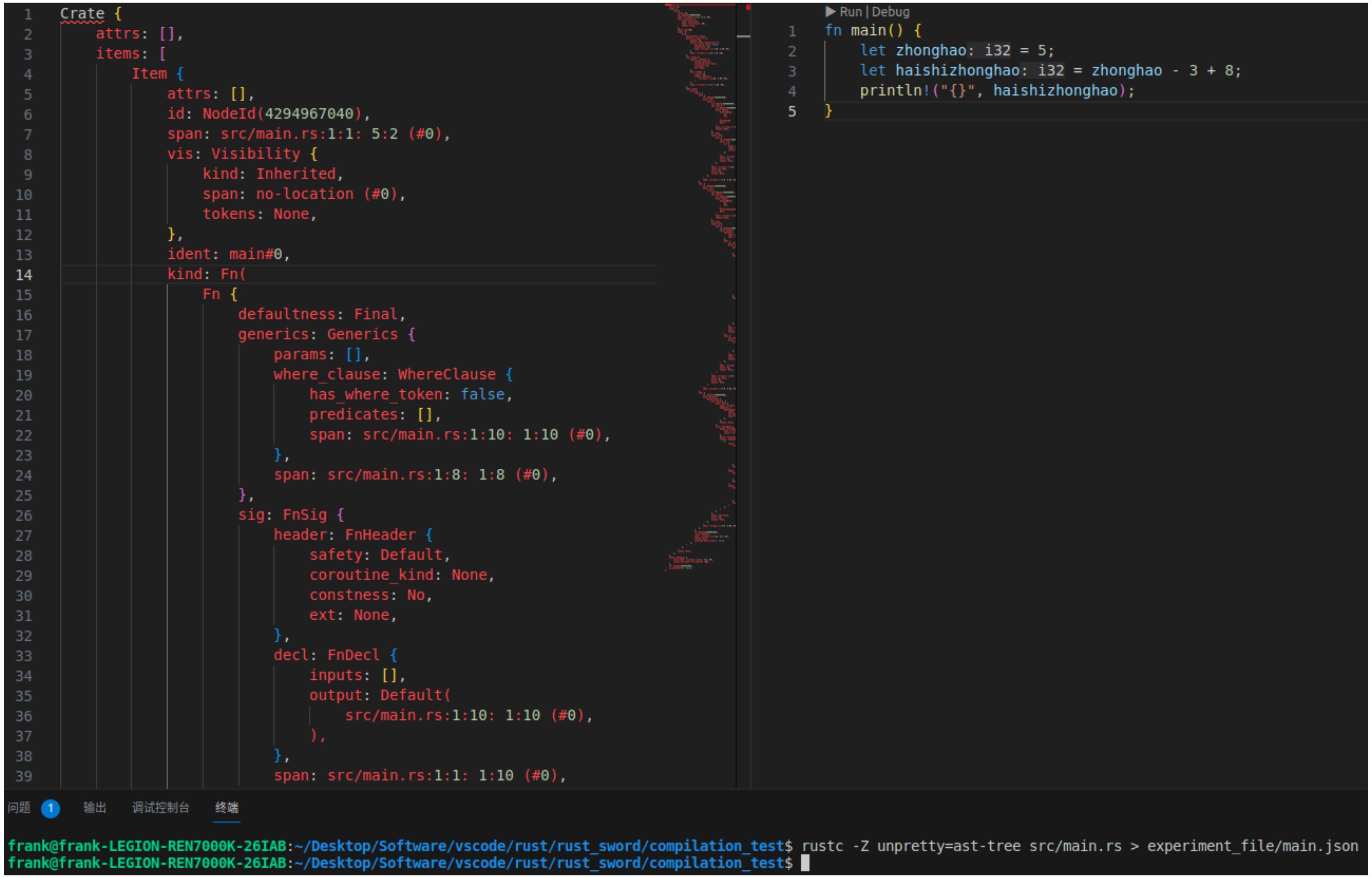
为什么要从语法树转回代码？

·生成代码很困难

·修改代码不容易

rust_ast_to_r：语法树转rust

Rust代码转语法树命令：rustc -Z unpretty=ast-tree src/main.rs > experiment_file/main.json

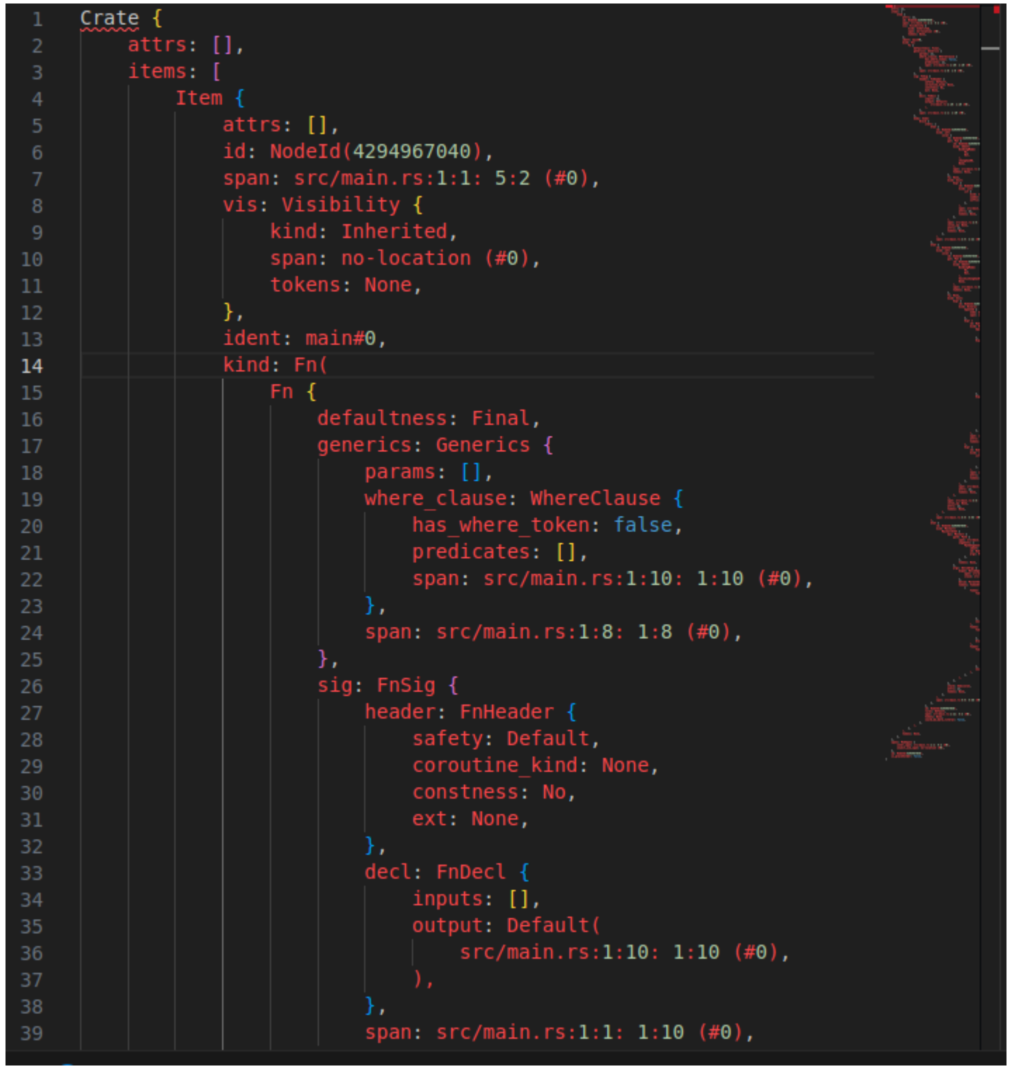


```
Crate {
    attrs: [],
    items: [
        Item {
            attrs: [],
            id: NodeId(4294967040),
            span: src/main.rs:1:1: 5:2 (#0),
            vis: Visibility {
                kind: Inherited,
                span: no-location (#0),
                tokens: None,
            },
            ident: main#0,
            kind: Fn(
                Fn {
                    defaultness: Final,
                    generics: Generics {
                        params: [],
                        where_clause: WhereClause {
                            has_where_token: false,
                            predicates: [],
                            span: src/main.rs:1:10: 1:10 (#0),
                        },
                        span: src/main.rs:1:8: 1:8 (#0),
                    },
                    sig: FnSig {
                        header: FnHeader {
                            safety: Default,
                            coroutine_kind: None,
                            constness: No,
                            ext: None,
                        },
                        decl: FnDecl {
                            inputs: [],
                            output: Default(
                                src/main.rs:1:10: 1:10 (#0),
                            ),
                        },
                        span: src/main.rs:1:1: 1:10 (#0),
```

```rust
fn main() {
    let zhonghao: i32 = 5;
    let haishizhonghao: i32 = zhonghao - 3 + 8;
    println!("{}", haishizhonghao);
}
```

```
frank@frank-LEGION-REN7000K-26IAB:~/Desktop/Software/vscode/rust/rust_sword/compilation_test$ rustc -Z unpretty=ast-tree src/main.rs > experiment_file/main.json
frank@frank-LEGION-REN7000K-26IAB:~/Desktop/Software/vscode/rust/rust_sword/compilation_test$ 
```

rust_ast_to_r：语法树转rust

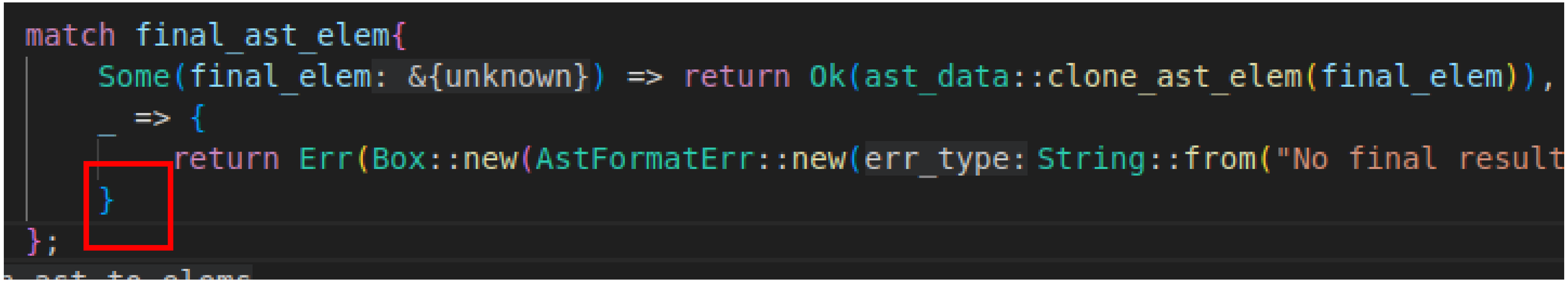


```
Crate {
    attrs: [],
    items: [
        Item {
            attrs: [],
            id: NodeId(4294967040),
            span: src/main.rs:1:1: 5:2 (#0),
            vis: Visibility {
                kind: Inherited,
                span: no-location (#0),
                tokens: None,
            },
            ident: main#0,
            kind: Fn(
                Fn {
                    defaultness: Final,
                    generics: Generics {
                        params: [],
                        where_clause: WhereClause {
                            has_where_token: false,
                            predicates: [],
                            span: src/main.rs:1:10: 1:10 (#0),
                        },
                        span: src/main.rs:1:8: 1:8 (#0),
                    },
                    sig: FnSig {
                        header: FnHeader {
                            safety: Default,
                            coroutine_kind: None,
                            constness: No,
                            ext: None,
                        },
                        decl: FnDecl {
                            inputs: [],
                            output: Default(
                                src/main.rs:1:10: 1:10 (#0),
                            ),
                        },
                        span: src/main.rs:1:1: 1:10 (#0),
```

从语法树转回rust代码：
能否转为书写相同的代码？
能否转为功能相同的代码？

rust_ast_to_r：语法树转rust

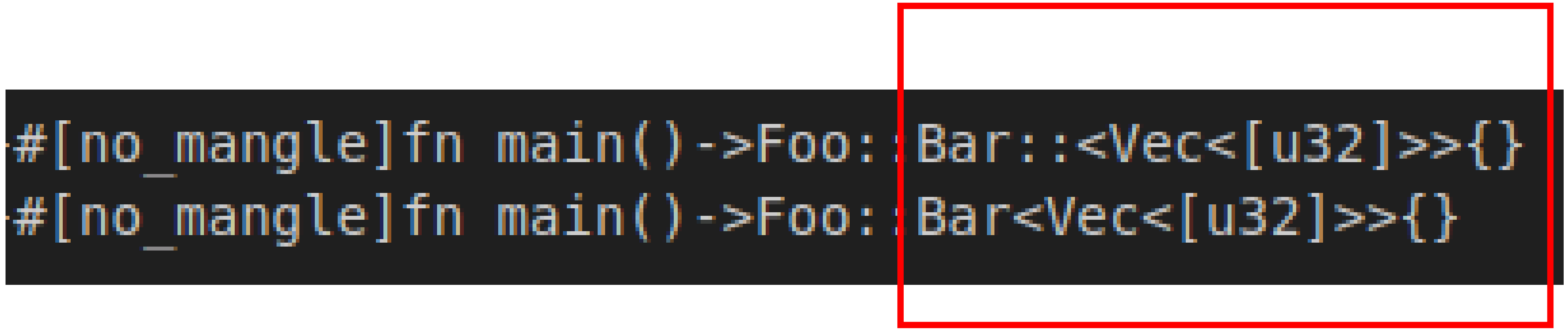
从语法树转回rust代码：
能否转为书写相同的代码？ ×



```
match final_ast_elem{
    Some(final_elem: &{unknown}) => return Ok(ast_data::clone_ast_elem(final_elem)),
    _ => {
        return Err(Box::new(AstFormatErr::new(err_type: String::from("No final result
    }
};
```

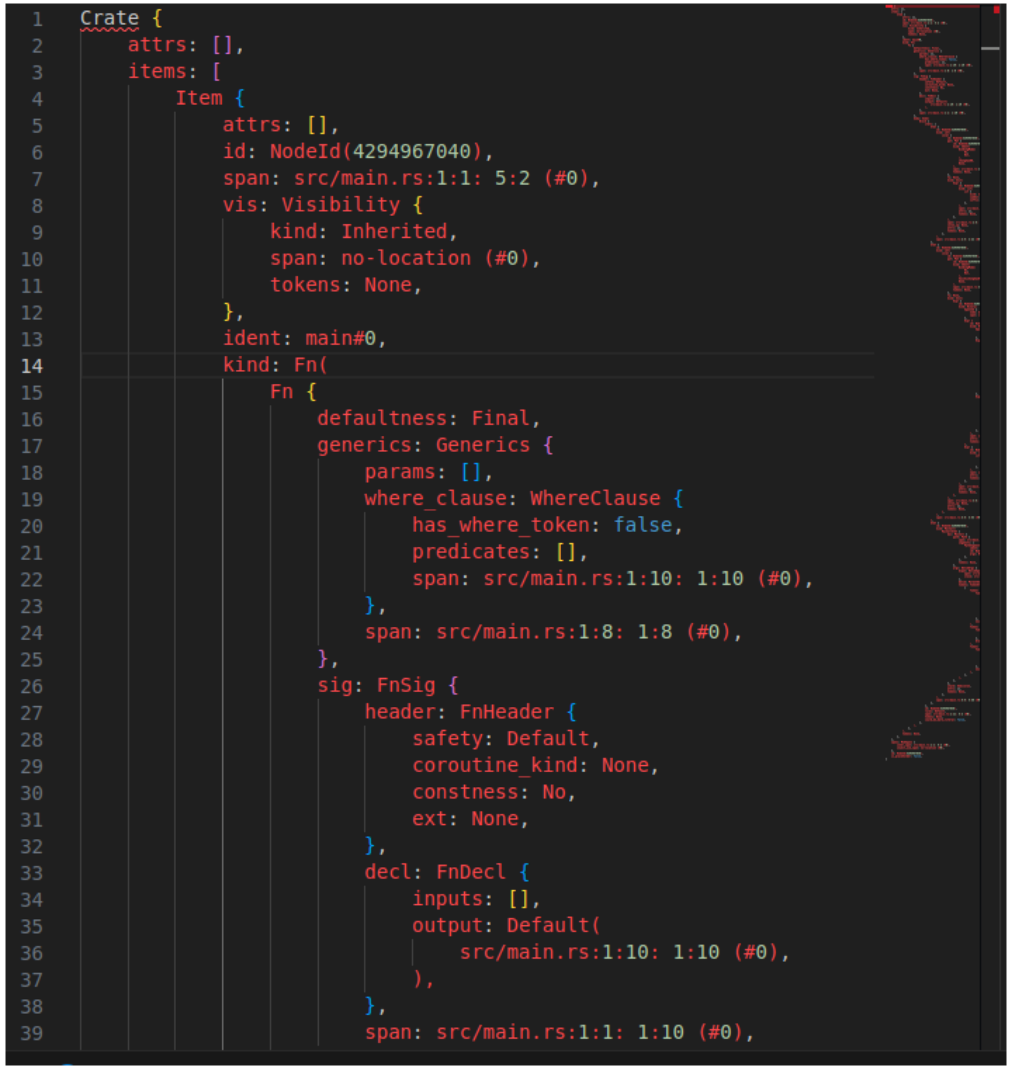
rust_ast_to_r：语法树转rust

从语法树转回rust代码:
能否转为功能相同的代码？



```
#[no_mangle]fn main()->Foo::Bar::<Vec<[u32]>>{}
#[no_mangle]fn main()->Foo::Bar<Vec<[u32]>>{}
```

rust_ast_to_r：语法树转rust



```
Crate {
    attrs: [],
    items: [
        Item {
            attrs: [],
            id: NodeId(4294967040),
            span: src/main.rs:1:1: 5:2 (#0),
            vis: Visibility {
                kind: Inherited,
                span: no-location (#0),
                tokens: None,
            },
            ident: main#0,
            kind: Fn(
                Fn {
                    defaultness: Final,
                    generics: Generics {
                        params: [],
                        where_clause: WhereClause {
                            has_where_token: false,
                            predicates: [],
                            span: src/main.rs:1:10: 1:10 (#0),
                        },
                        span: src/main.rs:1:8: 1:8 (#0),
                    },
                    sig: FnSig {
                        header: FnHeader {
                            safety: Default,
                            coroutine_kind: None,
                            constness: No,
                            ext: None,
                        },
                        decl: FnDecl {
                            inputs: [],
                            output: Default(
                                src/main.rs:1:10: 1:10 (#0),
                            ),
                        },
                        span: src/main.rs:1:1: 1:10 (#0),
```

rustc自带代码转语法树的工具
Sql有没有类似的语法树生成工具?
对于简单Sql，有没有别的方法?

Simple DB：语法树工具

Sql Parser：Rust库工具

node-sql-parser：支持 MySQL、PostgreSQL 的 SQL 解析为 AST。

PostgreSQL：使用 libpg_query（C 库，可提取 AST）

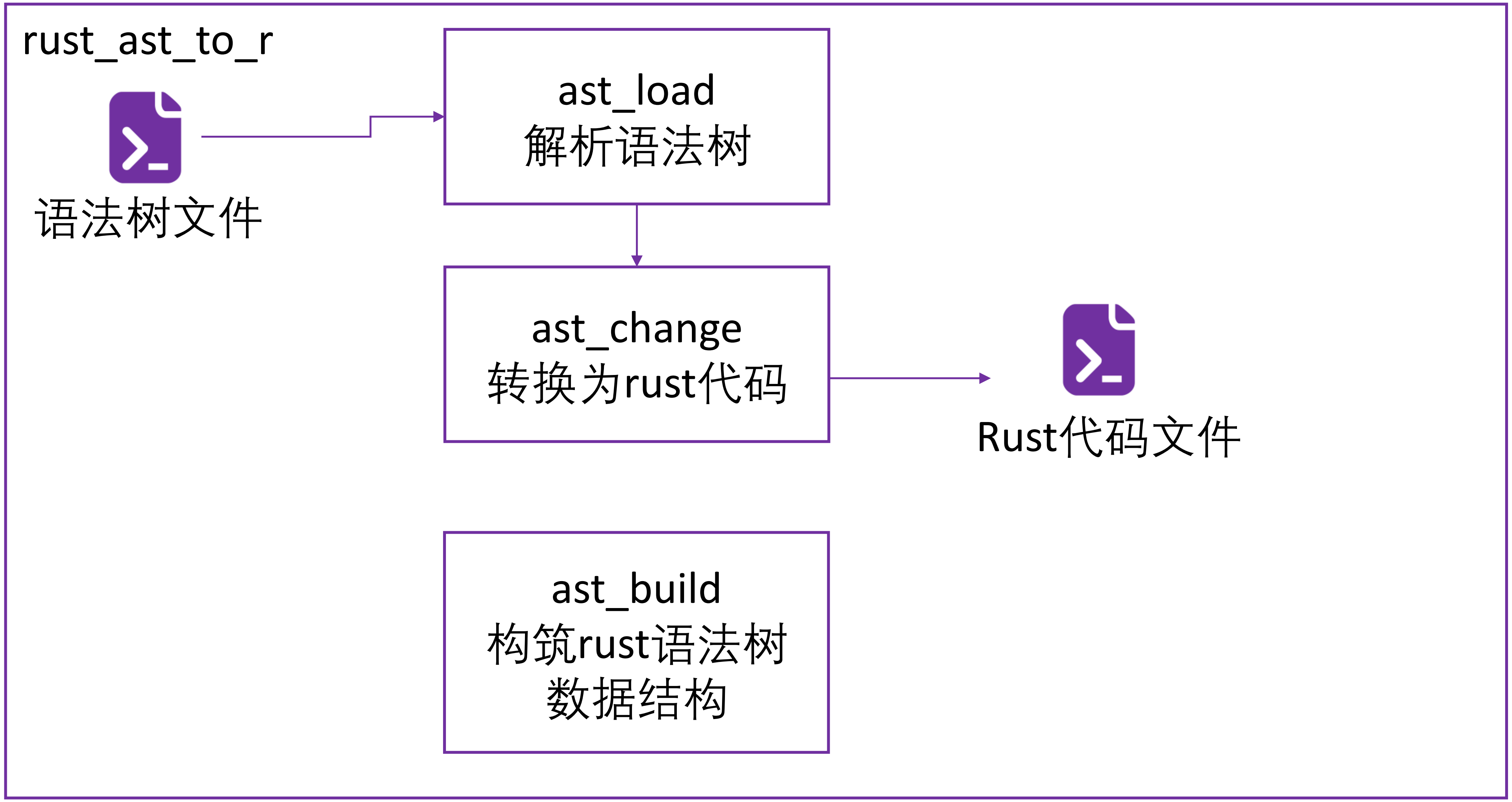
MySQL：mysql-server 源码中的 SQL 解析器（YACC 语法）

SQLite：sqlite3_prepare_v2 可生成语法树（VDBE 字节码前一步）

Simple DB: nom

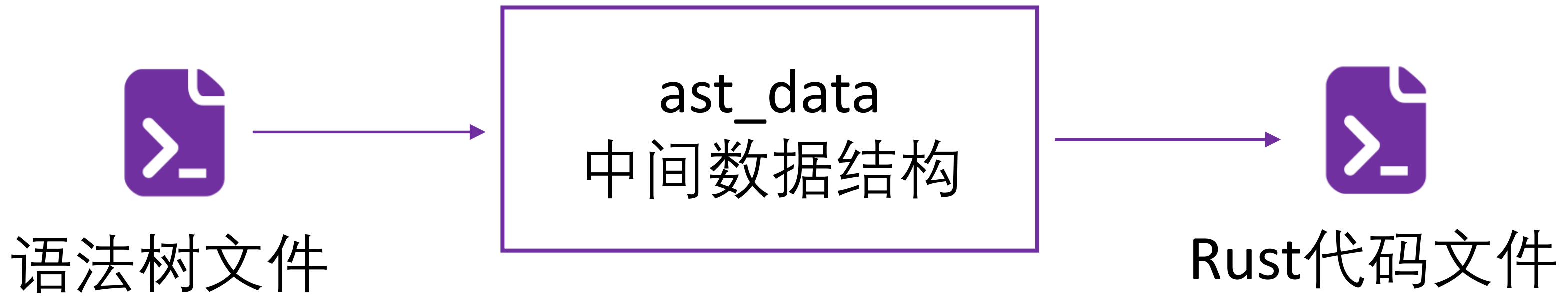
```rust
// IResult tracks the input type (generally string or bytes)
// and the output type for the parser
fn parser(s: &str) -> IResult<&str, &str> {
  tag_no_case("hello")(s)
}
// when you run a parser on some input it will return the remaining input in
the first param
// and the matched input for that parser that ran
assert_eq!(parser("Hello, World!"), Ok((", World!", "Hello")));
// NOTE: this is missing some trait bounds but the vibe is right
/// Run the given parser f on a comma seperated list
pub(crate) fn comma_sep<I, O, E, F>(
    f: F,
) -> impl FnMut(I) -> IResult<I, Vec<O>, E>
where
{
    separated_list1(tuple((multispace0, char(','), multispace0)), f)
}
```

rust_ast_to_r架构



rust_ast_to_r数据流

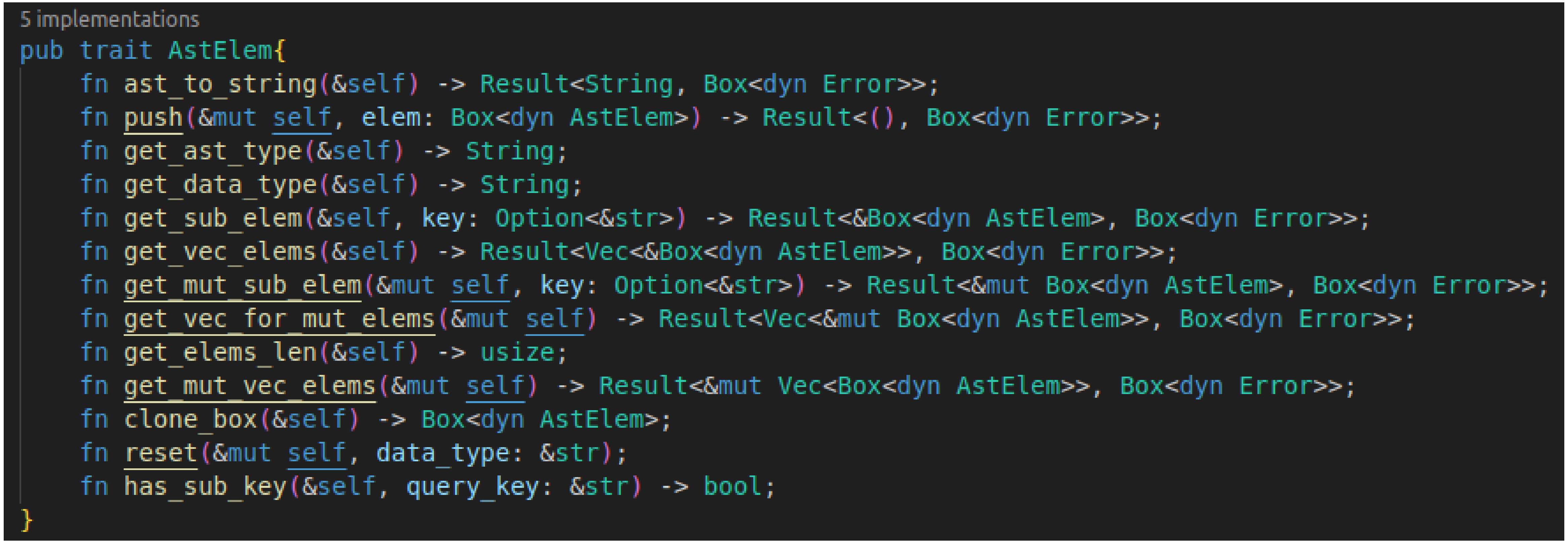
数据流



探讨rust_ast_to_r中的系统实践问题

ast_data

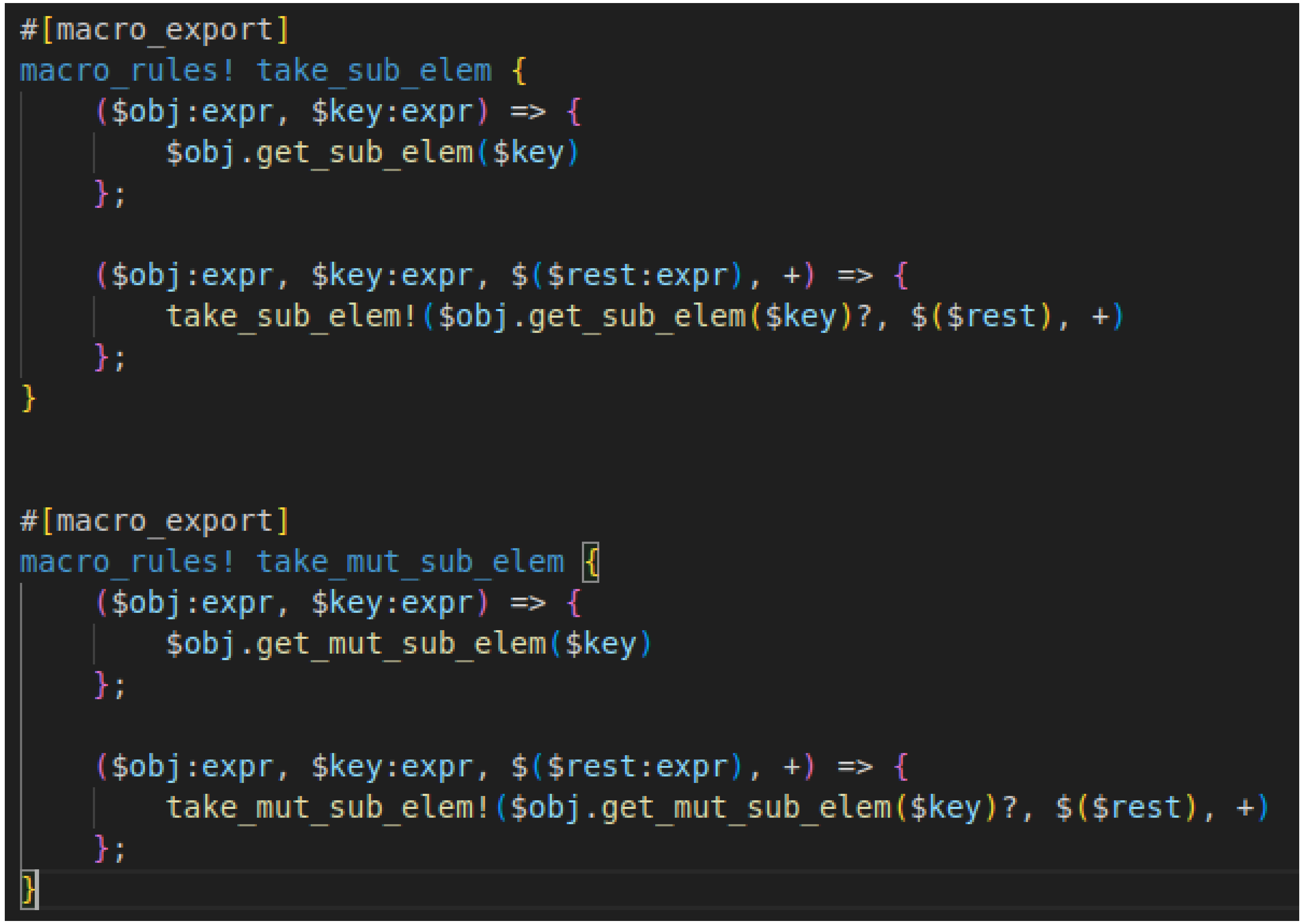
AstElem: BraceAstElem, PthAstElem, StrAstElem, SquareAstElem, StrAstElem



```
5 implementations
pub trait AstElem{
    fn ast_to_string(&self) -> Result<String, Box<dyn Error>>;
    fn push(&mut self, elem: Box<dyn AstElem>) -> Result<(), Box<dyn Error>>;
    fn get_ast_type(&self) -> String;
    fn get_data_type(&self) -> String;
    fn get_sub_elem(&self, key: Option<&str>) -> Result<&Box<dyn AstElem>, Box<dyn Error>>;
    fn get_vec_elems(&self) -> Result<Vec<&Box<dyn AstElem>>, Box<dyn Error>>;
    fn get_mut_sub_elem(&mut self, key: Option<&str>) -> Result<&mut Box<dyn AstElem>, Box<dyn Error>>;
    fn get_vec_for_mut_elems(&mut self) -> Result<Vec<&mut Box<dyn AstElem>>, Box<dyn Error>>;
    fn get_elems_len(&self) -> usize;
    fn get_mut_vec_elems(&mut self) -> Result<&mut Vec<Box<dyn AstElem>>, Box<dyn Error>>;
    fn clone_box(&self) -> Box<dyn AstElem>;
    fn reset(&mut self, data_type: &str);
    fn has_sub_key(&self, query_key: &str) -> bool;
}
```

探讨rust_ast_to_r中的系统实践问题

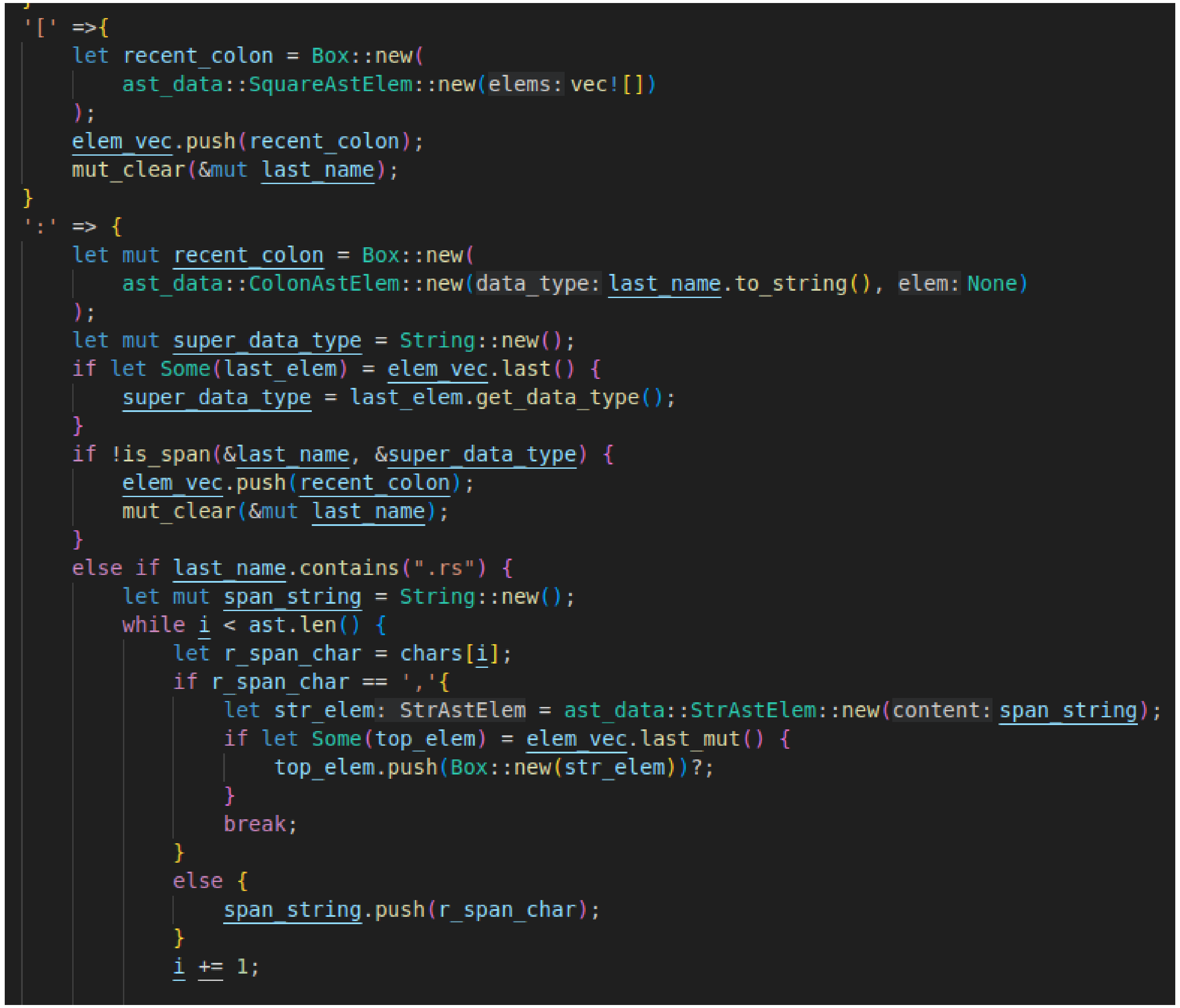
ast_data



```rust
#[macro_export]
macro_rules! take_sub_elem {
    ($obj:expr, $key:expr) => {
        $obj.get_sub_elem($key)
    };

    ($obj:expr, $key:expr, $($rest:expr), +) => {
        take_sub_elem!($obj.get_sub_elem($key)?, $($rest), +)
    };
}


#[macro_export]
macro_rules! take_mut_sub_elem {
    ($obj:expr, $key:expr) => {
        $obj.get_mut_sub_elem($key)
    };

    ($obj:expr, $key:expr, $($rest:expr), +) => {
        take_mut_sub_elem!($obj.get_mut_sub_elem($key)?, $($rest), +)
    };
}
```

探讨rust_ast_to_r中的系统实践问题

ast_load



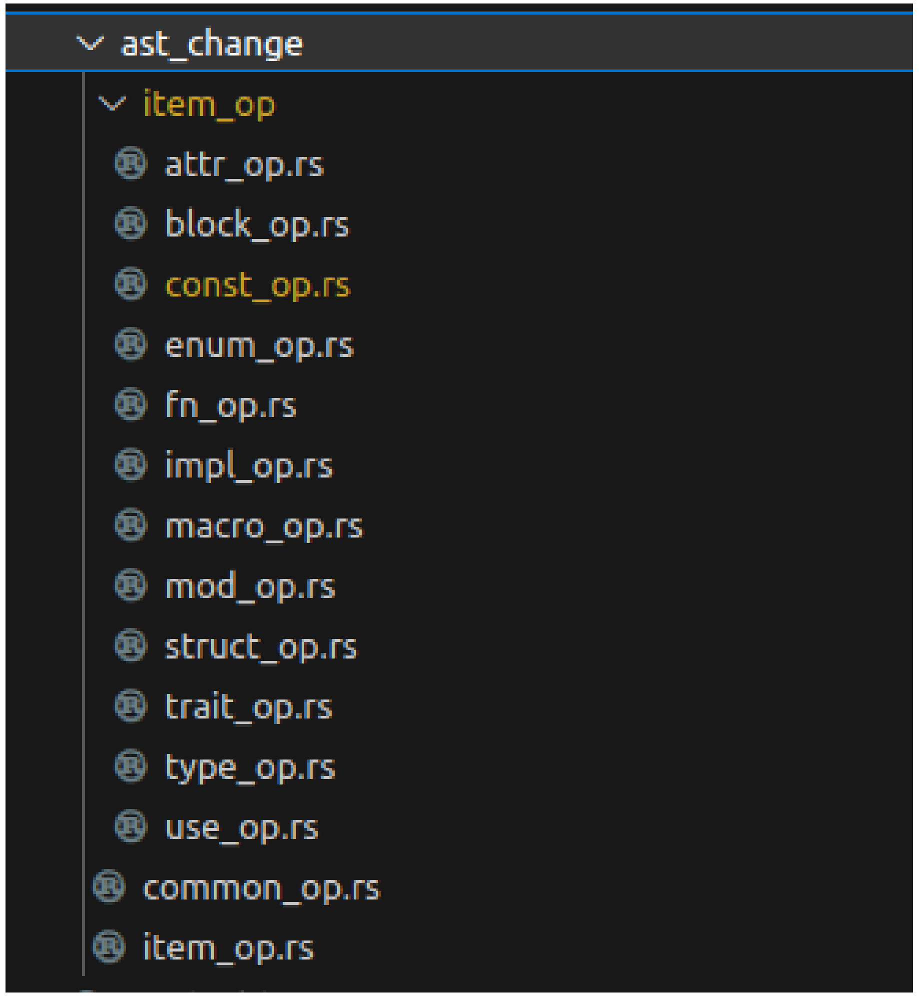
```
'[' =>{
    let recent_colon = Box::new(
        ast_data::SquareAstElem::new(elems: vec![])
    );
    elem_vec.push(recent_colon);
    mut_clear(&mut last_name);
}
':' => {
    let mut recent_colon = Box::new(
        ast_data::ColonAstElem::new(data_type: last_name.to_string(), elem: None)
    );
    let mut super_data_type = String::new();
    if let Some(last_elem) = elem_vec.last() {
        super_data_type = last_elem.get_data_type();
    }
    if !is_span(&last_name, &super_data_type) {
        elem_vec.push(recent_colon);
        mut_clear(&mut last_name);
    }
    else if last_name.contains(".rs") {
        let mut span_string = String::new();
        while i < ast.len() {
            let r_span_char = chars[i];
            if r_span_char == ','{
                let str_elem: StrAstElem = ast_data::StrAstElem::new(content: span_string);
                if let Some(top_elem) = elem_vec.last_mut() {
                    top_elem.push(Box::new(str_elem))?;
                }
                break;
            }
            else {
                span_string.push(r_span_char);
            }
            i += 1;
```

单向解析
优点：复杂度低
缺点：可读性差，不好维护

不太好的实现
根据实际情况
选择方法

探讨rust_ast_to_r中的系统实践问题

ast_change

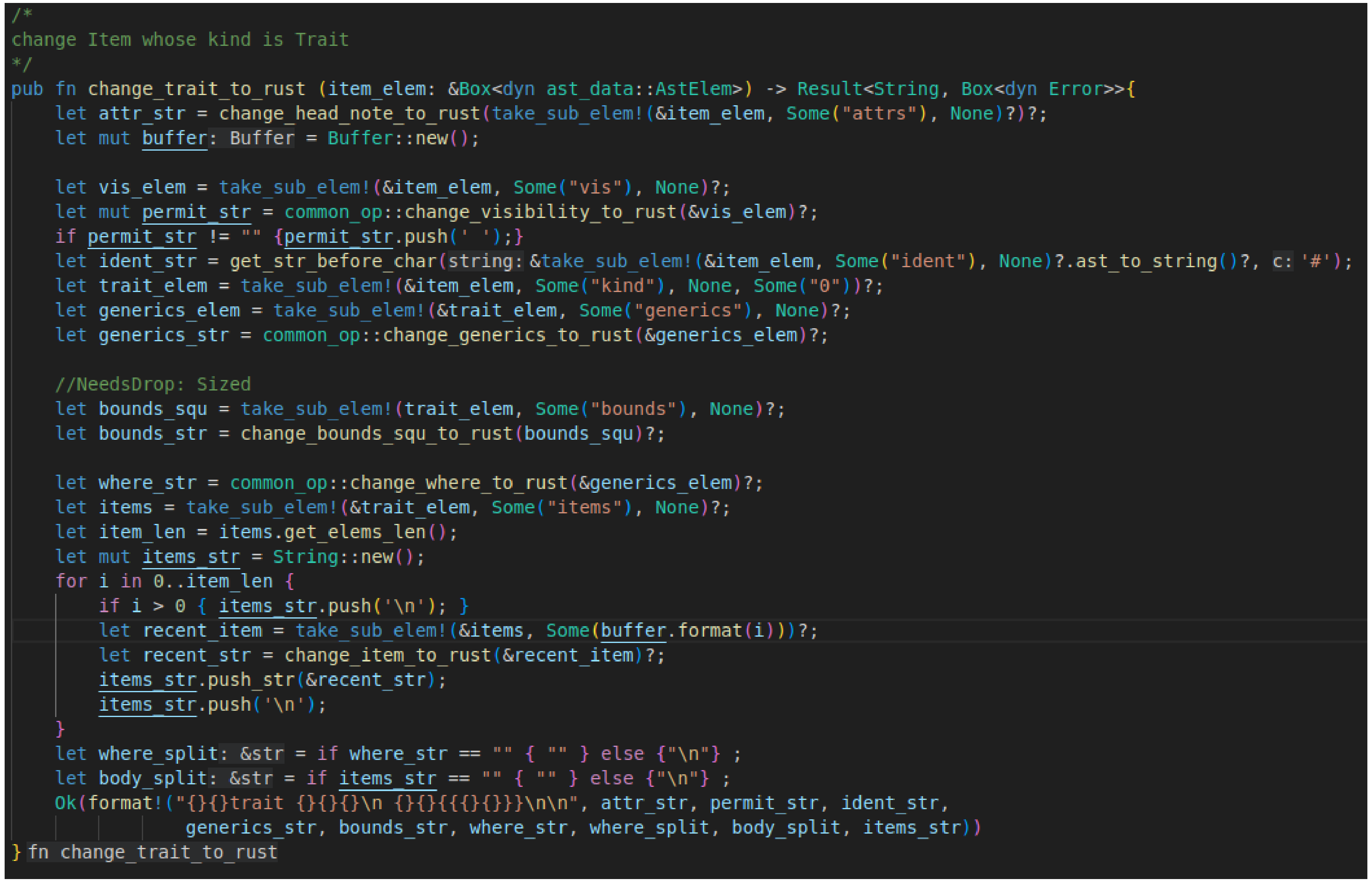


```
ast_change
  item_op
    attr_op.rs
    block_op.rs
    const_op.rs
    enum_op.rs
    fn_op.rs
    impl_op.rs
    macro_op.rs
    mod_op.rs
    struct_op.rs
    trait_op.rs
    type_op.rs
    use_op.rs
  common_op.rs
  item_op.rs
```

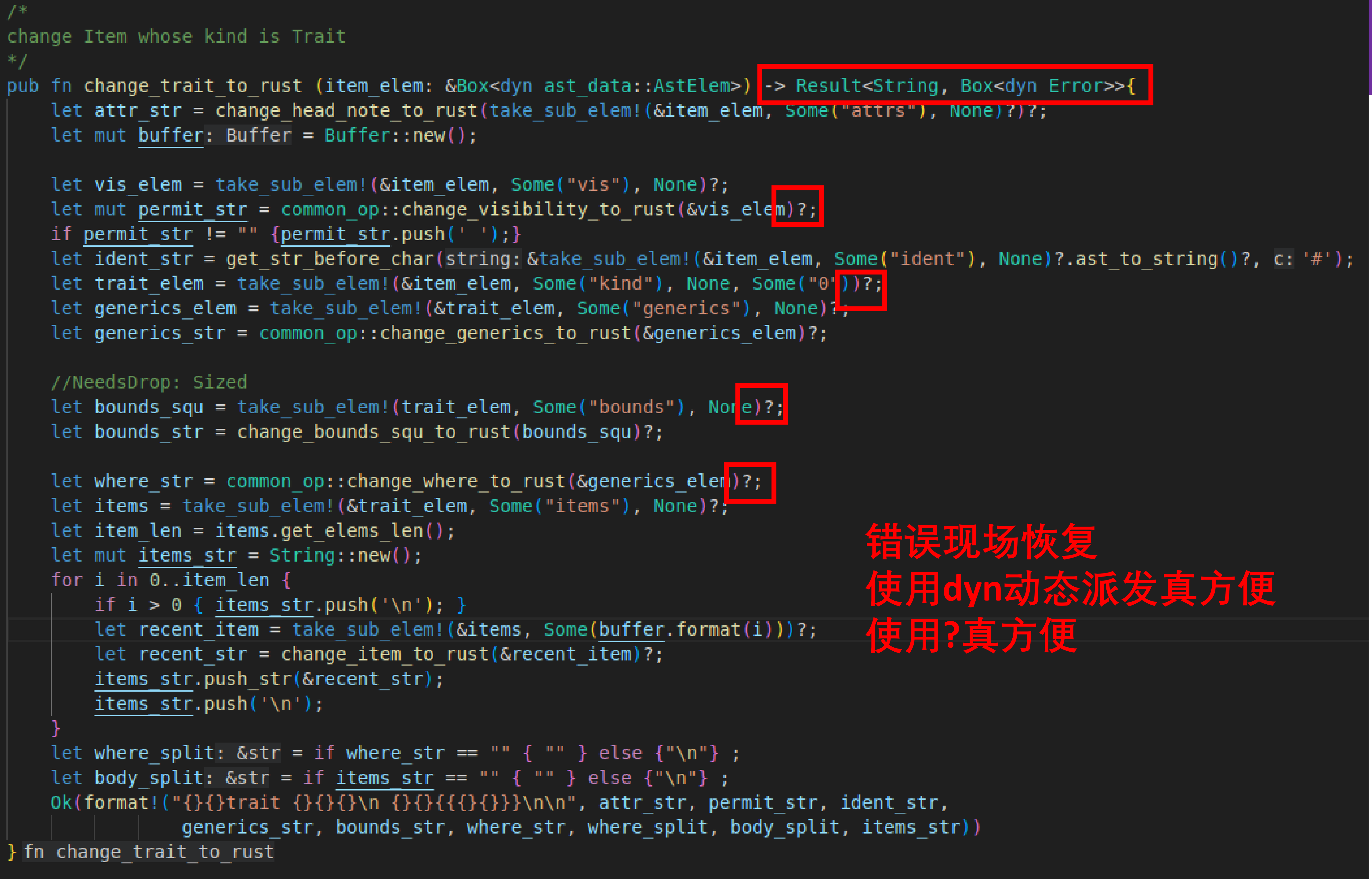
养成良好的目录
管理习惯

探讨rust_ast_to_r中的系统实践问题

ast_change： trait语法树转rust代码

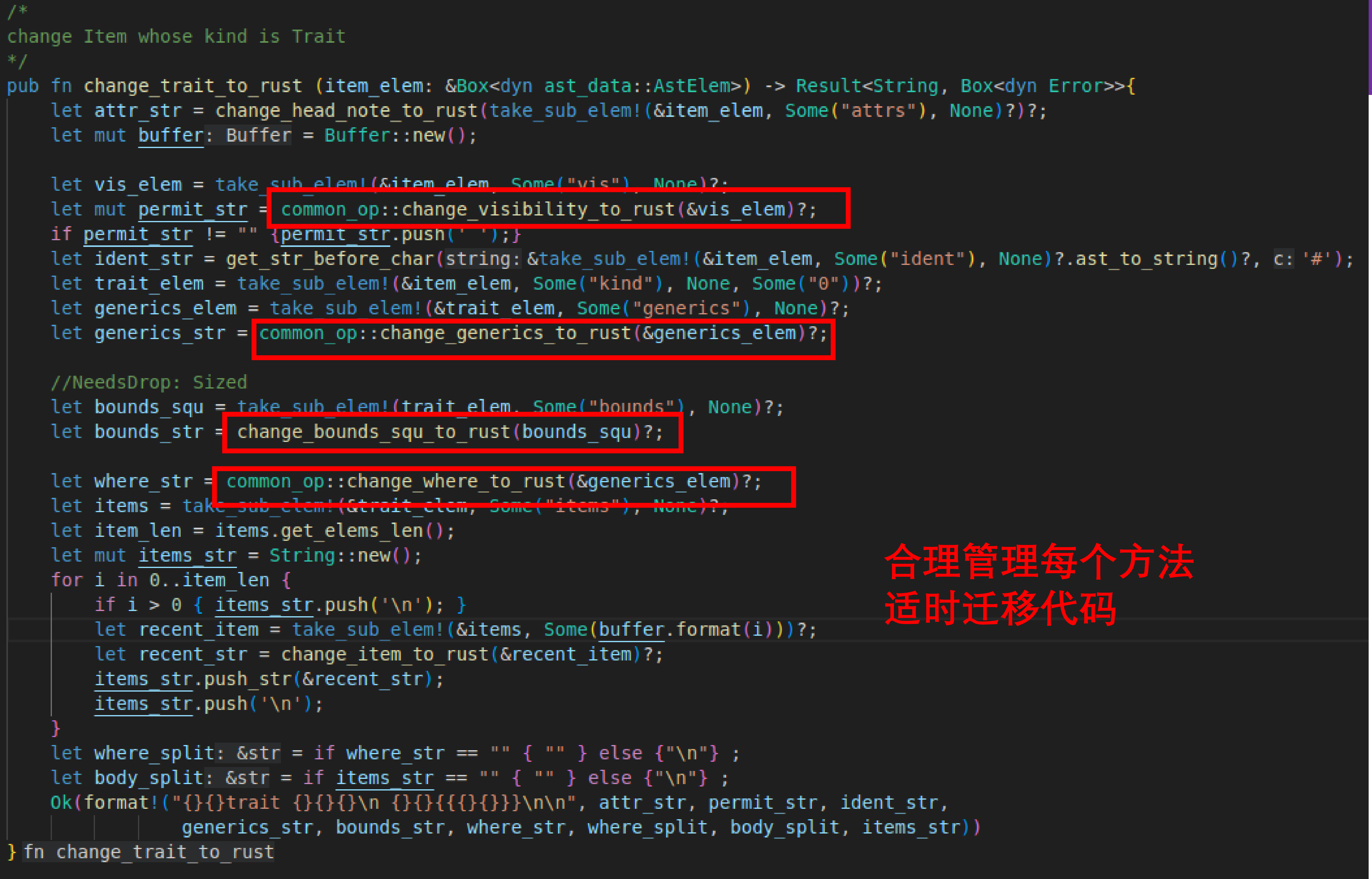


```rust
/*
change Item whose kind is Trait
*/
pub fn change_trait_to_rust (item_elem: &Box<dyn ast_data::AstElem>) -> Result<String, Box<dyn Error>>{
    let attr_str = change_head_note_to_rust(take_sub_elem!(&item_elem, Some("attrs"), None)?)?;
    let mut buffer: Buffer = Buffer::new();

    let vis_elem = take_sub_elem!(&item_elem, Some("vis"), None)?;
    let mut permit_str = common_op::change_visibility_to_rust(&vis_elem)?;
    if permit_str != "" {permit_str.push(' ');}
    let ident_str = get_str_before_char(string: &take_sub_elem!(&item_elem, Some("ident"), None)?.ast_to_string()?, c: '#');
    let trait_elem = take_sub_elem!(&item_elem, Some("kind"), None, Some("0"))?;
    let generics_elem = take_sub_elem!(&trait_elem, Some("generics"), None)?;
    let generics_str = common_op::change_generics_to_rust(&generics_elem)?;

    //NeedsDrop: Sized
    let bounds_squ = take_sub_elem!(trait_elem, Some("bounds"), None)?;
    let bounds_str = change_bounds_squ_to_rust(bounds_squ)?;

    let where_str = common_op::change_where_to_rust(&generics_elem)?;
    let items = take_sub_elem!(&trait_elem, Some("items"), None)?;
    let item_len = items.get_elems_len();
    let mut items_str = String::new();
    for i in 0..item_len {
        if i > 0 { items_str.push('\n'); }
        let recent_item = take_sub_elem!(&items, Some(buffer.format(i)))?;
        let recent_str = change_item_to_rust(&recent_item)?;
        items_str.push_str(&recent_str);
        items_str.push('\n');
    }
    let where_split: &str = if where_str == "" { "" } else {"\n"} ;
    let body_split: &str = if items_str == "" { "" } else {"\n"} ;
    Ok(format!("{}{}trait {}{}{}\n {}{}{{{}}}\n\n", attr_str, permit_str, ident_str,
            generics_str, bounds_str, where_str, where_split, body_split, items_str))
} fn change_trait_to_rust
```



```
/*
change Item whose kind is Trait
*/
pub fn change_trait_to_rust (item_elem: &Box<dyn ast_data::AstElem>) -> Result<String, Box<dyn Error>>{
    let attr_str = change_head_note_to_rust(take_sub_elem!(&item_elem, Some("attrs"), None)?)?;
    let mut buffer: Buffer = Buffer::new();

    let vis_elem = take_sub_elem!(&item_elem, Some("vis"), None)?;
    let mut permit_str = common_op::change_visibility_to_rust(&vis_elem)?;
    if permit_str != "" {permit_str.push(' ');}
    let ident_str = get_str_before_char(string: &take_sub_elem!(&item_elem, Some("ident"), None)?.ast_to_string()?, c: '#');
    let trait_elem = take_sub_elem!(&item_elem, Some("kind"), None, Some("0"))?;
    let generics_elem = take_sub_elem!(&trait_elem, Some("generics"), None)?;
    let generics_str = common_op::change_generics_to_rust(&generics_elem)?;

    //NeedsDrop: Sized
    let bounds_squ = take_sub_elem!(trait_elem, Some("bounds"), None)?;
    let bounds_str = change_bounds_squ_to_rust(bounds_squ)?;

    let where_str = common_op::change_where_to_rust(&generics_elem)?;
    let items = take_sub_elem!(&trait_elem, Some("items"), None)?;
    let item_len = items.get_elems_len();
    let mut items_str = String::new();
    for i in 0..item_len {
        if i > 0 { items_str.push('\n'); }
        let recent_item = take_sub_elem!(&items, Some(buffer.format(i)))?;
        let recent_str = change_item_to_rust(&recent_item)?;
        items_str.push_str(&recent_str);
        items_str.push('\n');
    }
    let where_split: &str = if where_str == "" { "" } else {"\n"} ;
    let body_split: &str = if items_str == "" { "" } else {"\n"} ;
    Ok(format!("{}{}trait {}{}{}\n {}{}{{{}{}}}\n\n", attr_str, permit_str, ident_str,
                generics_str, bounds_str, where_str, where_split, body_split, items_str))
} fn change_trait_to_rust
```

错误现场恢复
使用dyn动态派发真方便
使用?真方便

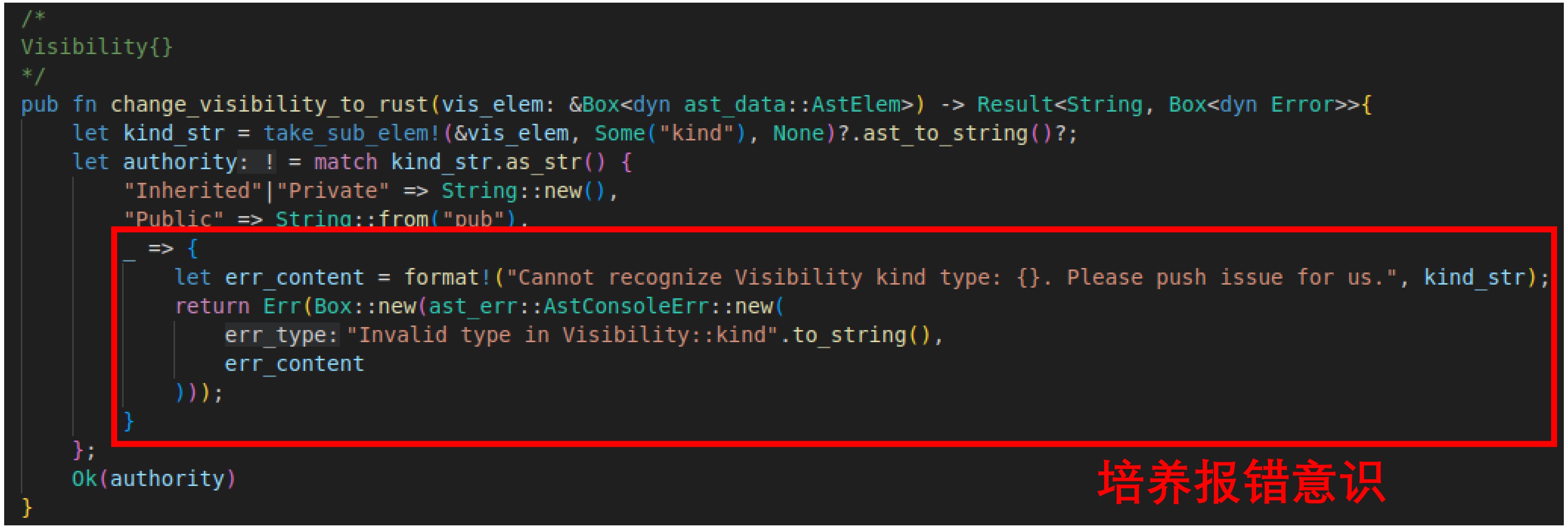


```
/*
change Item whose kind is Trait
*/
pub fn change_trait_to_rust (item_elem: &Box<dyn ast_data::AstElem>) -> Result<String, Box<dyn Error>>{
    let attr_str = change_head_note_to_rust(take_sub_elem!(&item_elem, Some("attrs"), None)?)?;
    let mut buffer: Buffer = Buffer::new();

    let vis_elem = take_sub_elem!(&item_elem, Some("vis"), None)?;
    let mut permit_str = common_op::change_visibility_to_rust(&vis_elem)?;
    if permit_str != "" {permit_str.push(' ');}
    let ident_str = get_str_before_char(string: &take_sub_elem!(&item_elem, Some("ident"), None)?.ast_to_string()?, c: '#');
    let trait_elem = take_sub_elem!(&item_elem, Some("kind"), None, Some("0"))?;
    let generics_elem = take_sub_elem!(&trait_elem, Some("generics"), None)?;
    let generics_str = common_op::change_generics_to_rust(&generics_elem)?;

    //NeedsDrop: Sized
    let bounds_squ = take_sub_elem!(trait_elem, Some("bounds"), None)?;
    let bounds_str = change_bounds_squ_to_rust(bounds_squ)?;

    let where_str = common_op::change_where_to_rust(&generics_elem)?;
    let items = take_sub_elem!(&trait_elem, Some("items"), None)?;
    let item_len = items.get_elems_len();
    let mut items_str = String::new();
    for i in 0..item_len {
        if i > 0 { items_str.push('\n'); }
        let recent_item = take_sub_elem!(&items, Some(buffer.format(i)))?;
        let recent_str = change_item_to_rust(&recent_item)?;
        items_str.push_str(&recent_str);
        items_str.push('\n');
    }
    let where_split: &str = if where_str == "" { "" } else {"\n"} ;
    let body_split: &str = if items_str == "" { "" } else {"\n"} ;
    Ok(format!("{}{}trait {}{}{}\n {}{}{{{}{}}}\n\n", attr_str, permit_str, ident_str,
                generics_str, bounds_str, where_str, where_split, body_split, items_str))
} fn change_trait_to_rust
```

合理管理每个方法
适时迁移代码

探讨rust_ast_to_r中的系统实践问题

ast_change

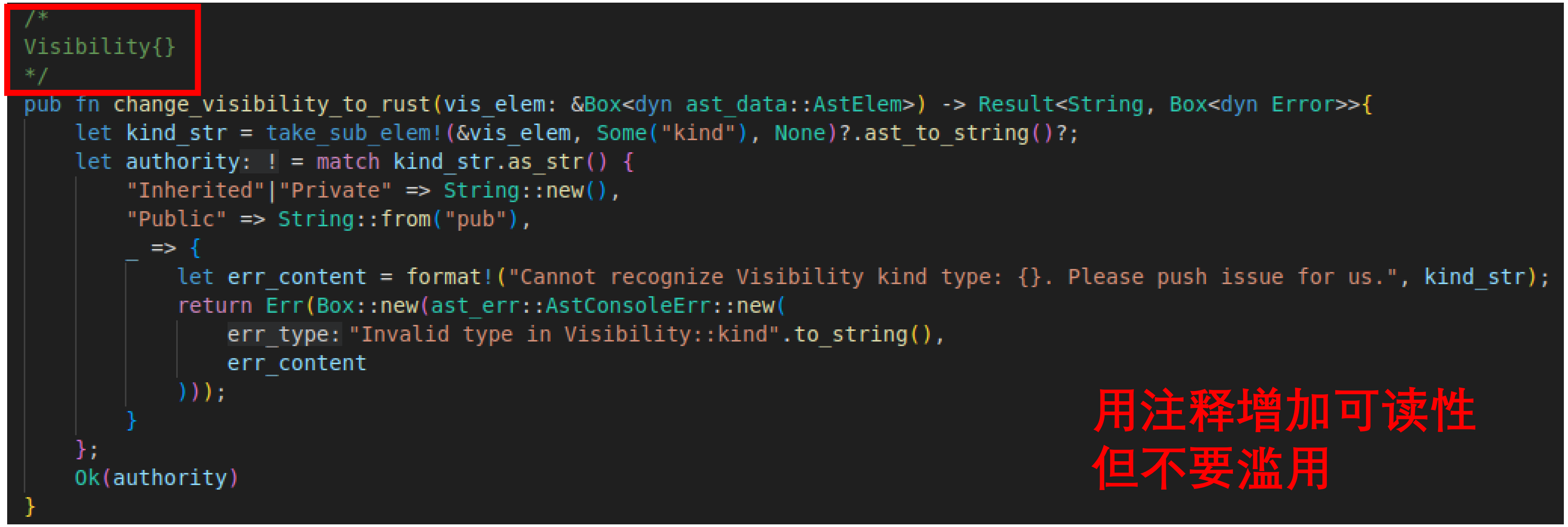


```
/*
Visibility{}
*/
pub fn change_visibility_to_rust(vis_elem: &Box<dyn ast_data::AstElem>) -> Result<String, Box<dyn Error>>{
    let kind_str = take_sub_elem!(&vis_elem, Some("kind"), None)?.ast_to_string()?;
    let authority: ! = match kind_str.as_str() {
        "Inherited"|"Private" => String::new(),
        "Public" => String::from("pub"),
        _ => {
            let err_content = format!("Cannot recognize Visibility kind type: {}. Please push issue for us.", kind_str);
            return Err(Box::new(ast_err::AstConsoleErr::new(
                err_type: "Invalid type in Visibility::kind".to_string(),
                err_content
            )));
        }
    };
    Ok(authority)
}
```

培养报错意识

探讨rust_ast_to_r中的系统实践问题

ast_change



```
/*
Visibility{}
*/
pub fn change_visibility_to_rust(vis_elem: &Box<dyn ast_data::AstElem>) -> Result<String, Box<dyn Error>>{
    let kind_str = take_sub_elem!(&vis_elem, Some("kind"), None)?.ast_to_string()?;
    let authority: ! = match kind_str.as_str() {
        "Inherited"|"Private" => String::new(),
        "Public" => String::from("pub"),
        _ => {
            let err_content = format!("Cannot recognize Visibility kind type: {}. Please push issue for us.", kind_str);
            return Err(Box::new(ast_err::AstConsoleErr::new(
                err_type: "Invalid type in Visibility::kind".to_string(),
                err_content
            )));
        }
    };
    Ok(authority)
}
```

用注释增加可读性
但不要滥用

敬请各位同学批评指正！

汇报人：潘中颢

南京大学软件学院

Email: zhonghaopan@smail.nju.edu.cn