(Maybe) Owning Things, Declaratively

Modelling Rust's Ownership in Datalog

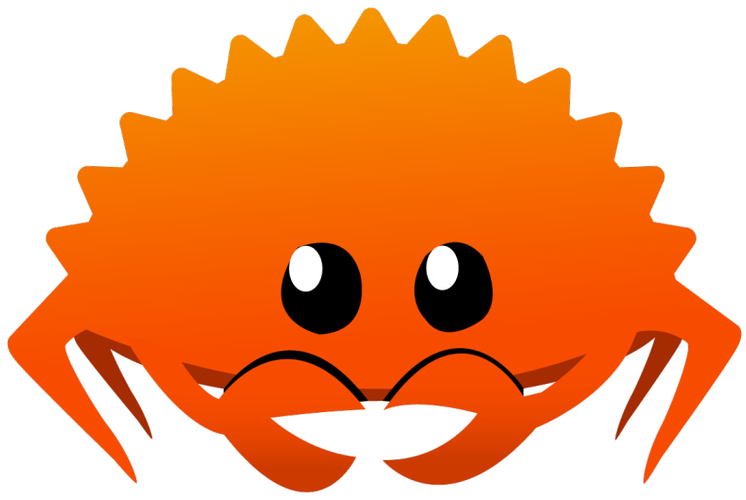
Who?

- I am Albin, and this is my Master's project
- Working with Niko Matsakis (et. al.) of Mozilla Research
- Tobias is subject reviewing

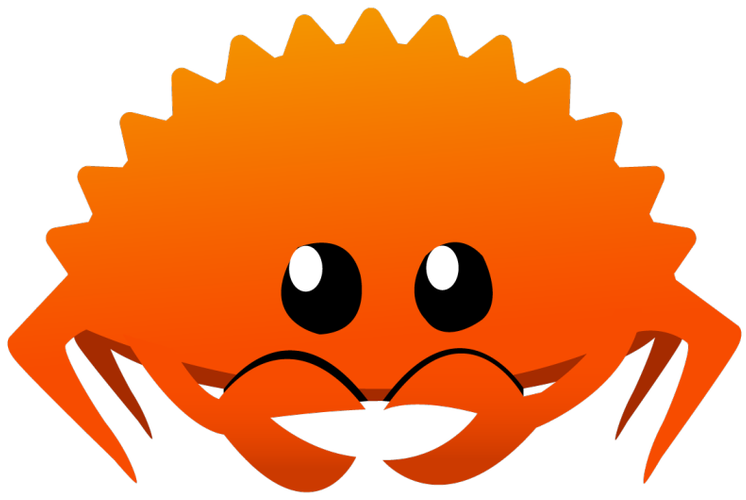
Who?

- I am Albin, and this is my Master's project
- Working with Niko Matsakis (et. al.) of Mozilla Research
- Tobias is subject reviewing

What?

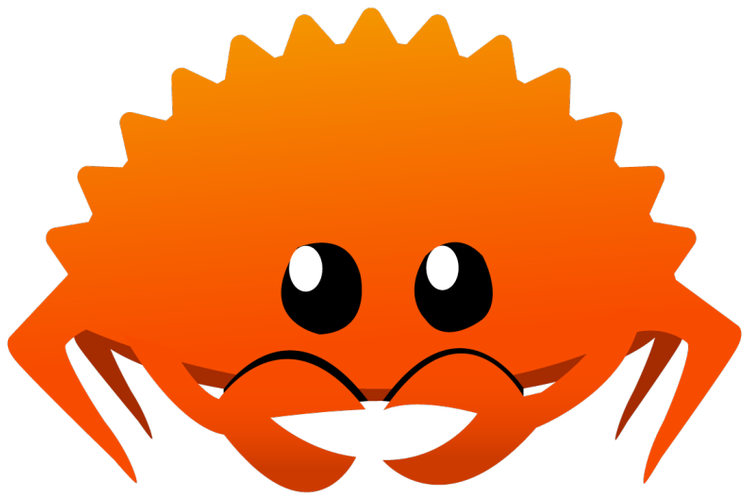


What?



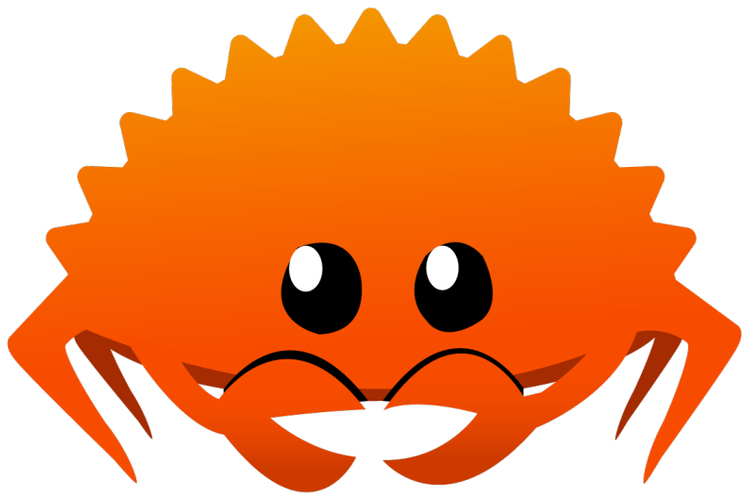
What?

- **Polonius**: a declarative reformulation of the Rust borrow check



What?

- **Polonius**: a declarative reformulation of the Rust borrow check
- Separate package (“crate”) consumed by Rust

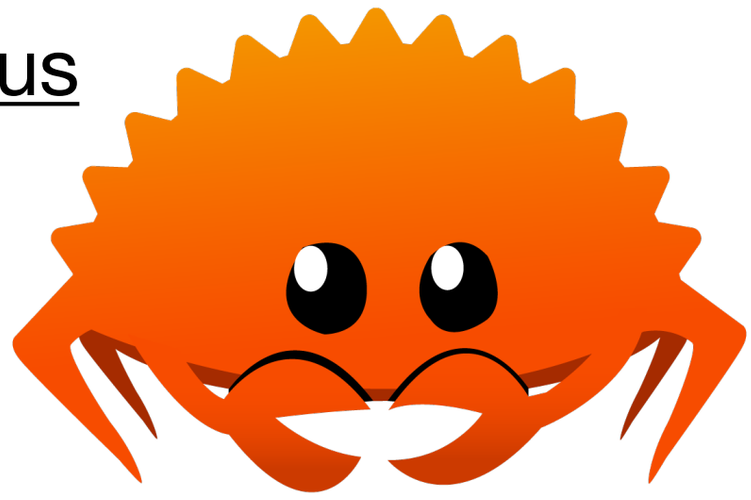


What?

- **Polonius**: a declarative reformulation of the Rust borrow check

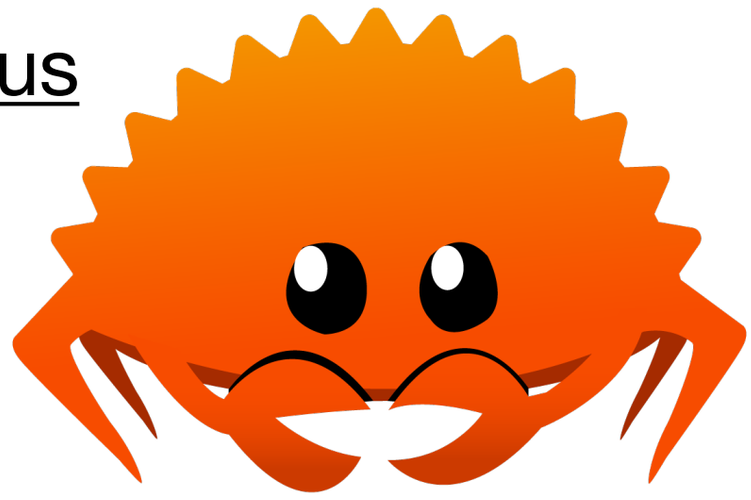
- Separate package (“crate”) consumed by Rust

- Developed on GitHub: rust-lang/polonius



What?

- **Polonius**: a declarative reformulation of the Rust borrow check
- Separate package (“crate”) consumed by Rust
- Developed on GitHub: rust-lang/polonius
- Current status: *very* work-in-progress!



What??

What??

- The borrow check: static analysis of references

What??

- The borrow check: static analysis of references
- Used to ensure safe shared memory use

What??

- The borrow check: static analysis of references
- Used to ensure safe shared memory use
- May-point-to (“provenance”) part of a reference’s type

What??

- The borrow check: static analysis of references
- Used to ensure safe shared memory use
- May-point-to (“provenance”) part of a reference’s type
  - How do we interpret such a type? Stay tuned!

What???

What???

House rules

What???

House rules

- A value must be (initialised | not moved) when used

What???

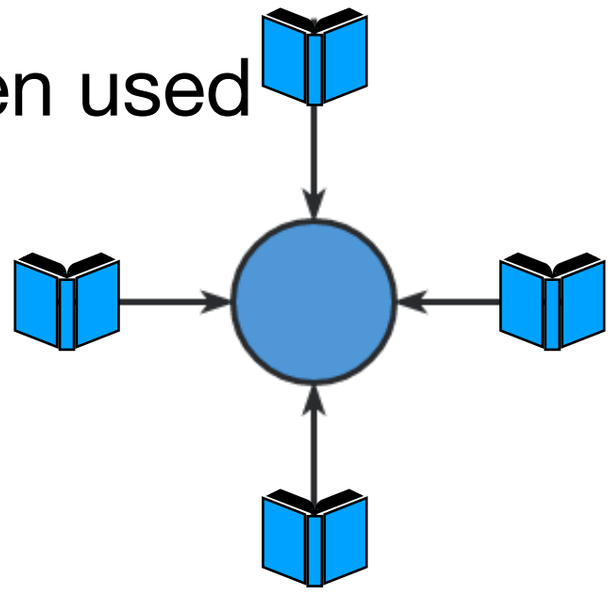
House rules

- A value must be (initialised | not moved) when used
- *N* (live) read-references to same value **OR**

What???

House rules

- A value must be (initialised | not moved) when used

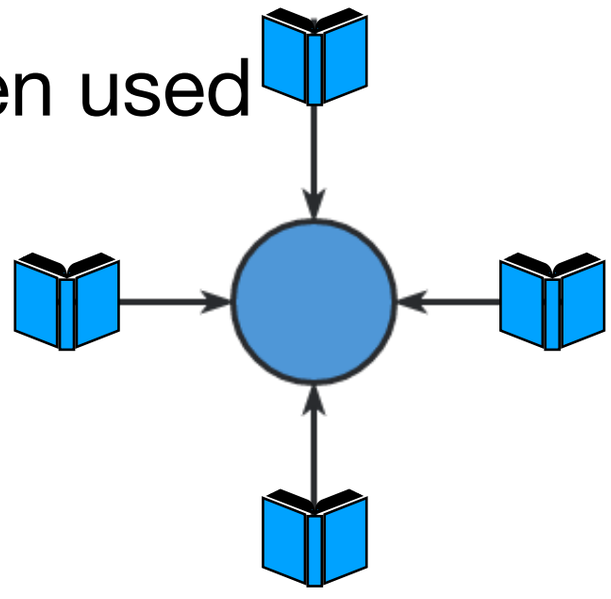


- *N* (live) read-references to same value **OR**

What???

House rules

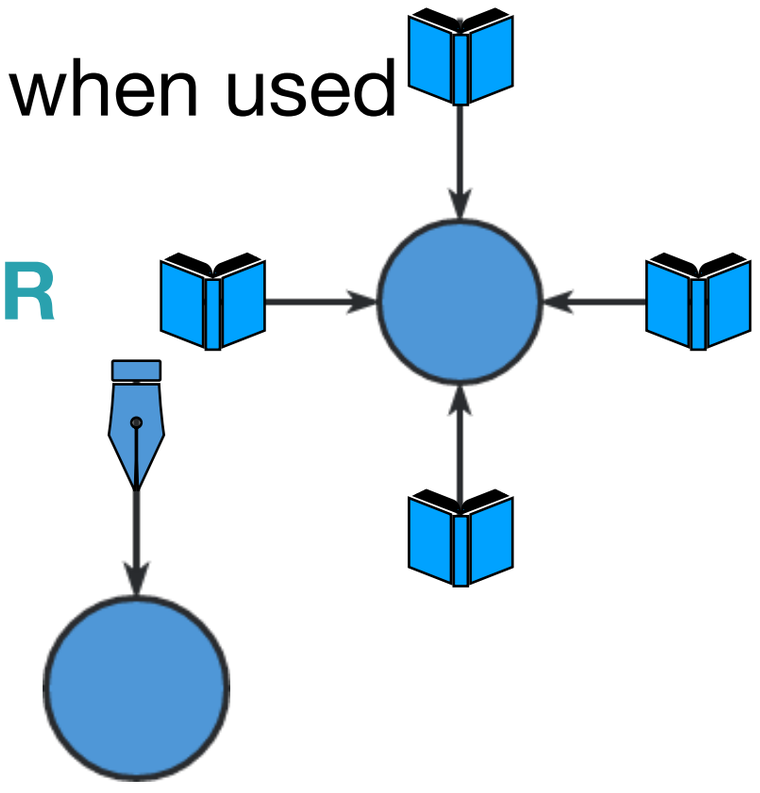
- A value must be (initialised | not moved) when used
- *N* (live) read-references to same value **OR**
- ≤ 1 (live) write-reference to same value



What???

House rules

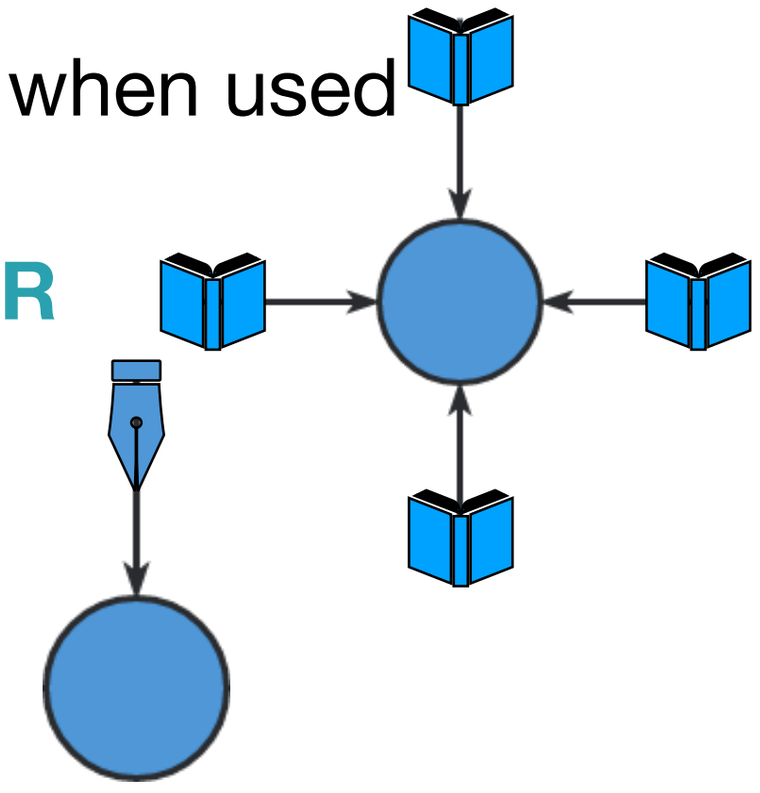
- A value must be (initialised | not moved) when used
- *N* (live) read-references to same value **OR**
- ≤ 1 (live) write-reference to same value



What???

House rules

- A value must be (initialised | not moved) when used
- *N* (live) read-references to same value **OR**
- ≤ 1 (live) write-reference to same value
- A reference must never outlive its value



What????

```
let mut x = 22;
let y = 44;
let mut p = &x; // L0: x is borrowed
```

```
p = &y; // L1: y is borrowed, overwrites p


x += 1; // OK, because p was overwritten
println!(*p); // 44
```

Why?

Why?

- Easier development and experimentation

Why?

- Easier development and experimentation
- Common logic *implemented once*

Why?

- Easier development and experimentation
- Common logic *implemented once*
- Similar logic *looks similar*

Why?

- Easier development and experimentation
- Common logic *implemented once*
- Similar logic *looks similar*
- Promising Performance in the Literature™

Why?

- Easier development and experimentation
- Common logic *implemented once*
- Similar logic *looks similar*
- Promising Performance in the Literature™
- More advanced analysis

How?

```
let mut x = 22;
let y = 44;
let mut p: &'a i32 = &'b x; // L0

p = &'c y; // L1


x += 1;

println!(*p);
```

How?

```
let mut x = 22;
let y = 44;
let mut p: &'a i32 = &'b x; // L0
p = &'c y; // L1

x += 1;

println!(*p);
```

Provenance variables

How?

```
let mut x = 22;
let y = 44;
let mut p: &'a i32 = &'b x; // L0

p = &'c y; // L1


x += 1;

println!(*p);
```

How?

```
let mut x = 22;
let y = 44;
let mut p: &'a i32 = &'b x; // L0

p = &'c y; // L1


x += 1;

println!(*p);
```

```
'b ⊆ 'a
{L0} ⊆ 'b
```

How?

```
let mut x = 22;
let y = 44;
let mut p: &'a i32 = &'b x; // L0


p = &'c y; // L1


x += 1;

println!(*p);
```

'b ⊆ 'a
{L0} ⊆ 'b

'c ⊆ 'a
{L1} ⊆ 'c
'a: O, I am slain! ☠

How?

```
let mut x = 22;
let y = 44;
let mut p: &'a i32 = &'b x; // L0

p = &'c y; // L1

x += 1;

println!(*p);
```

'b ⊆ 'a
{L0} ⊆ 'b

'c ⊆ 'a
{L1} ⊆ 'c
'a: 0, I am slain! ☠️

invalidates(L0)

How??

How??

```
provvar_live_at(R, P) :-
    var_drop_live(V, P),
    var_drops_provvar(V, R).
```

```
provvar_live_at(R, P) :-
    var_live(V, P),
    var_uses_provvar(V, R).

var_live(V, P) :-
    var_live(V, Q),
    cfg_edge(P, Q),
    !var_defined(V, P).
```

How??

```
provvar_live_at(R, P) :-
    var_drop_live(V, P),
    var_drops_provvar(V, R).

provvar_live_at(R, P) :-
    var_live(V, P),
    var_uses_provvar(V, R).

var_live(V, P) :-
    var_live(V, Q),
    cfg_edge(P, Q),
    !var_defined(V, P).
```

```
provvar_live_at.from_join(
  &var_drop_live,
  &var_drops_provvar, |_v, &p, &r| {
  ((r, p), ())
});

provvar_live_at.from_join(
  &var_live_var,
  &var_uses_provvar, |_v, &p, &r| {
  ((r, p), ())
});

var_live_var.from_leapjoin(
  &var_live_var,
  (
  defined_rel.extend_anti(
      |&(v, _q)| v),
  edge_reverse_cfg.extend_with(
      |&(_v, q)| q),
  ),
  |&(v, _q), &p| (v, p),
);
```

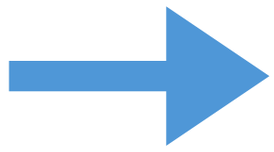
How???

How???

```rust
fn foo() {
    let x = 22;
    let p = &x;
}
```

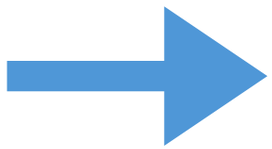
How???

```
fn foo() {
    let x = 22;
    let p = &x;
}
```



How???

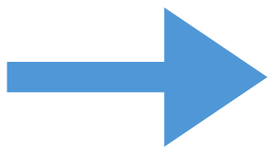
```
fn foo() {
    let x = 22;
    let p = &x;
}
```



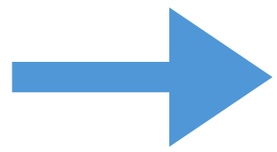
```
bb0: {
    StorageLive(_1);
    _1 = const 22i32;
    FakeRead(ForLet, _1);
    StorageLive(_2);
    _2 = &_1;
    FakeRead(ForLet, _2);
    _0 = ();
    StorageDead(_2);
    StorageDead(_1);
    return;
}
```

How???

```
fn foo() {
    let x = 22;
    let p = &x;
}
```

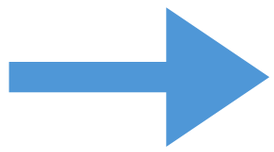


```
bb0: {
    StorageLive(_1);
    _1 = const 22i32;
    FakeRead(ForLet, _1);
    StorageLive(_2);
    _2 = &_1;
    FakeRead(ForLet, _2);
    _0 = ();
    StorageDead(_2);
    StorageDead(_1);
    return;
}
```

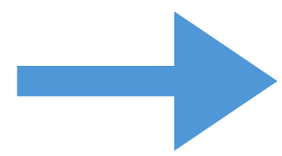


How???

```
fn foo() {
    let x = 22;
    let p = &x;
}
```

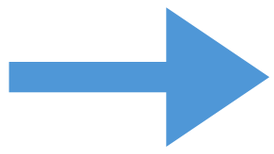


```
bb0: {
    StorageLive(_1);
    _1 = const 22i32;
    FakeRead(ForLet, _1);
    StorageLive(_2);
    _2 = &_1;
    FakeRead(ForLet, _2);
    _0 = ();
    StorageDead(_2);
    StorageDead(_1);
    return;
}
```

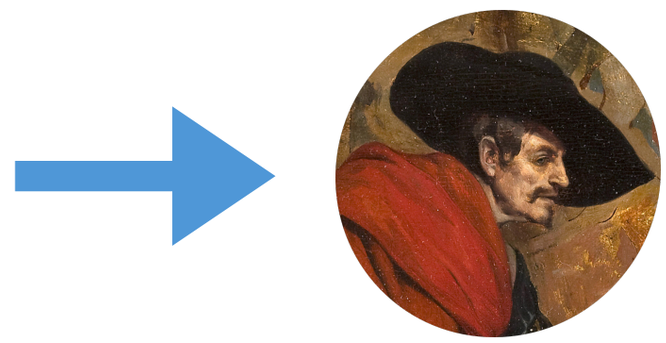


How???

```
fn foo() {
    let x = 22;
    let p = &x;
}
```



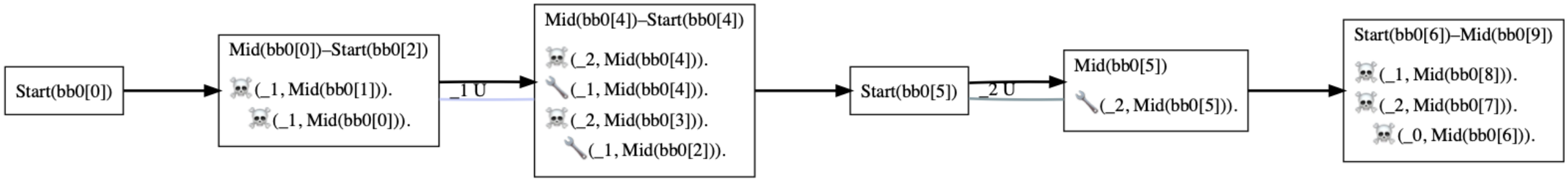
```
bb0: {
    StorageLive(_1);
    _1 = const 22i32;
    FakeRead(ForLet, _1);
    StorageLive(_2);
    _2 = &_1;
    FakeRead(ForLet, _2);
    _0 = ();
    StorageDead(_2);
    StorageDead(_1);
    return;
}
```



How???

```
fn foo() {
    let x = 22;
    let p = &x;
}
```

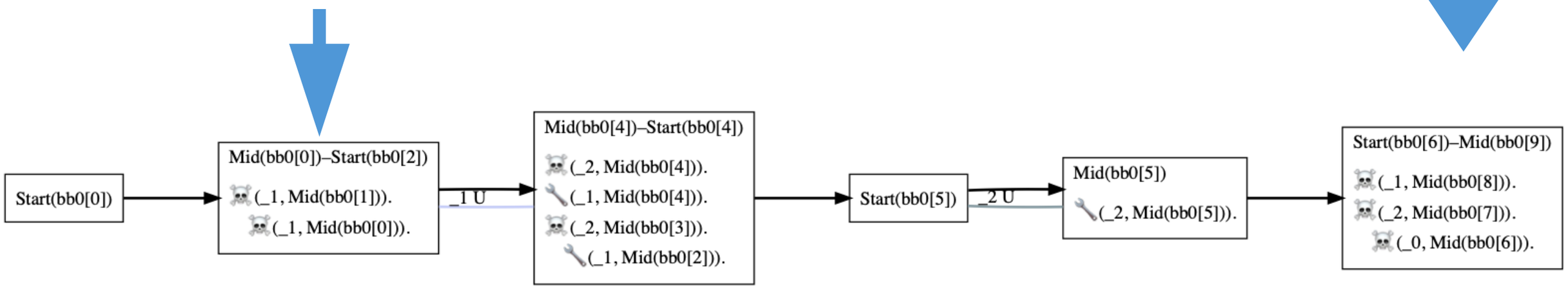
```
bb0: {
    StorageLive(_1);
    _1 = const 22i32;
    FakeRead(ForLet, _1);
    StorageLive(_2);
    _2 = &_1;
    FakeRead(ForLet, _2);
    _0 = ();
    StorageDead(_2);
    StorageDead(_1);
    return;
}
```



How???

```
fn foo() {
    let x = 22;
    let p = &x;
}
```

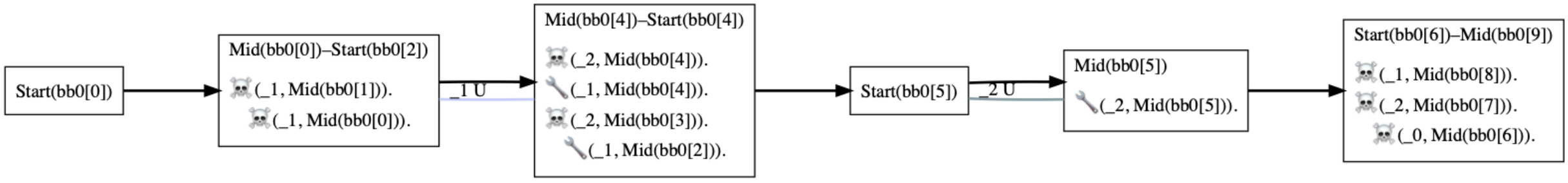
```
bb0: {
    StorageLive(_1);
    _1 = const 22i32;
    FakeRead(ForLet, _1);
    StorageLive(_2);
    _2 = &_1;
    FakeRead(ForLet, _2);
    _0 = ();
    StorageDead(_2);
    StorageDead(_1);
    return;
}
```



How???

```
fn foo() {
    let x = 22;
    let p = &x;
}
```

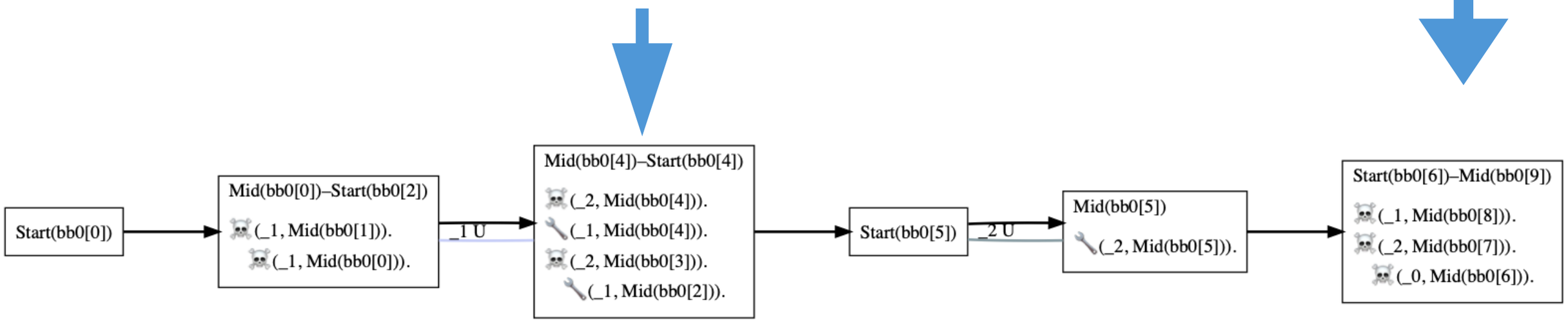
```
bb0: {
    StorageLive(_1);
    _1 = const 22i32;
    FakeRead(ForLet, _1);
    StorageLive(_2);
    _2 = &_1;
    FakeRead(ForLet, _2);
    _0 = ();
    StorageDead(_2);
    StorageDead(_1);
    return;
}
```



How???

```
fn foo() {
    let x = 22;
    let p = &x;
}
```

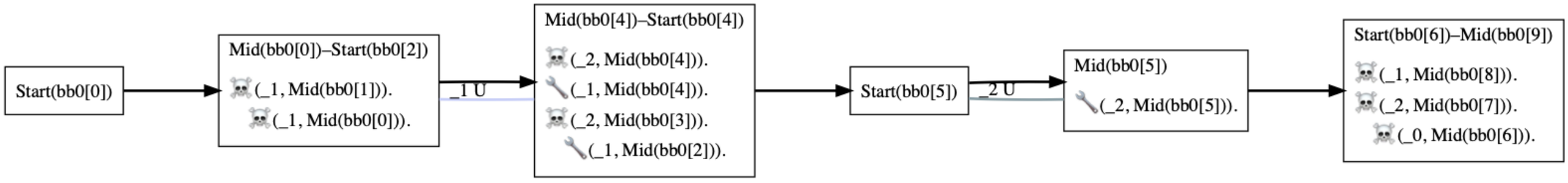
```
bb0: {
    StorageLive(_1);
    _1 = const 22i32;
    FakeRead(ForLet, _1);
    StorageLive(_2);
    _2 = &_1;
    FakeRead(ForLet, _2);
    _0 = ();
    StorageDead(_2);
    StorageDead(_1);
    return;
}
```



How???

```
fn foo() {
    let x = 22;
    let p = &x;
}
```

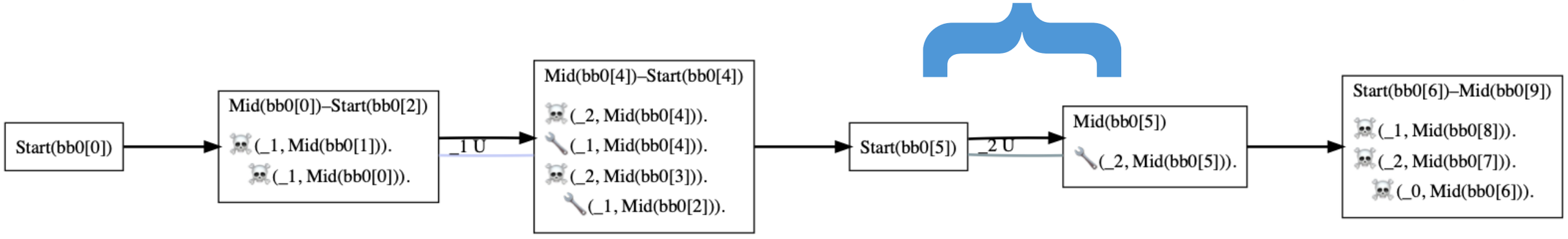
```
bb0: {
    StorageLive(_1);
    _1 = const 22i32;
    FakeRead(ForLet, _1);
    StorageLive(_2);
    _2 = &_1;
    FakeRead(ForLet, _2);
    _0 = ();
    StorageDead(_2);
    StorageDead(_1);
    return;
}
```



How???

```
fn foo() {
    let x = 22;
    let p = &x;
}
```

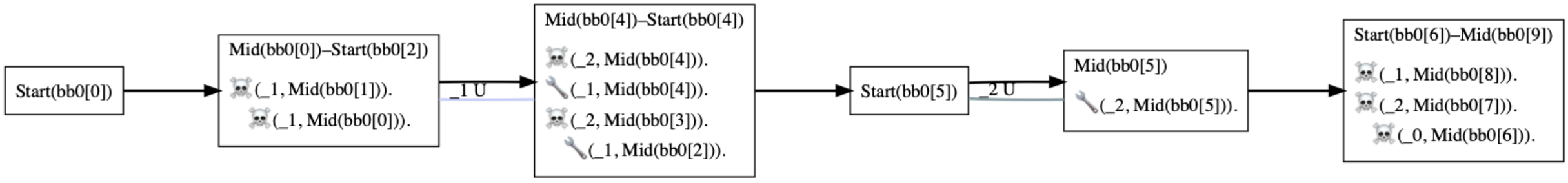
```
bb0: {
    StorageLive(_1);
    _1 = const 22i32;
    FakeRead(ForLet, _1);
    StorageLive(_2);
    _2 = &_1;
    FakeRead(ForLet, _2);
    _0 = ();
    StorageDead(_2);
    StorageDead(_1);
    return;
}
```



How???

```
fn foo() {
    let x = 22;
    let p = &x;
}
```

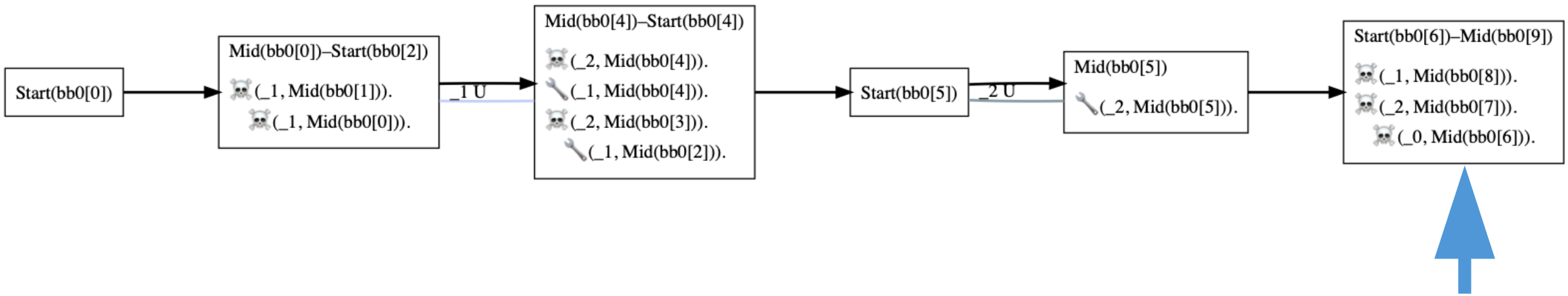
```
bb0: {
    StorageLive(_1);
    _1 = const 22i32;
    FakeRead(ForLet, _1);
    StorageLive(_2);
    _2 = &_1;
    FakeRead(ForLet, _2);
    _0 = ();
    StorageDead(_2);
    StorageDead(_1);
    return;
}
```



How???

```
fn foo() {
    let x = 22;
    let p = &x;
}
```

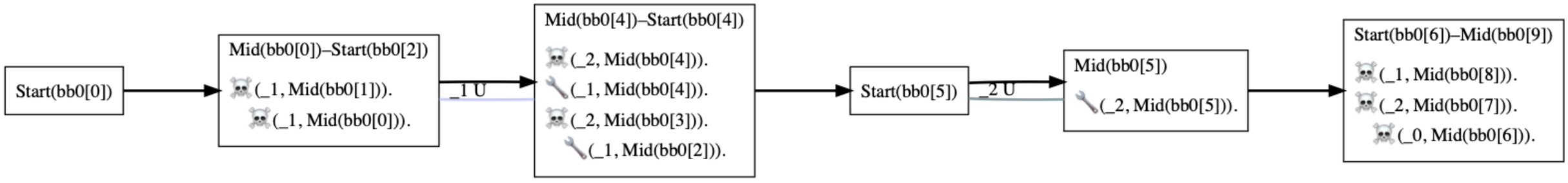
```
bb0: {
    StorageLive(_1);
    _1 = const 22i32;
    FakeRead(ForLet, _1);
    StorageLive(_2);
    _2 = &_1;
    FakeRead(ForLet, _2);
    _0 = ();
    StorageDead(_2);
    StorageDead(_1);
    return;
}
```

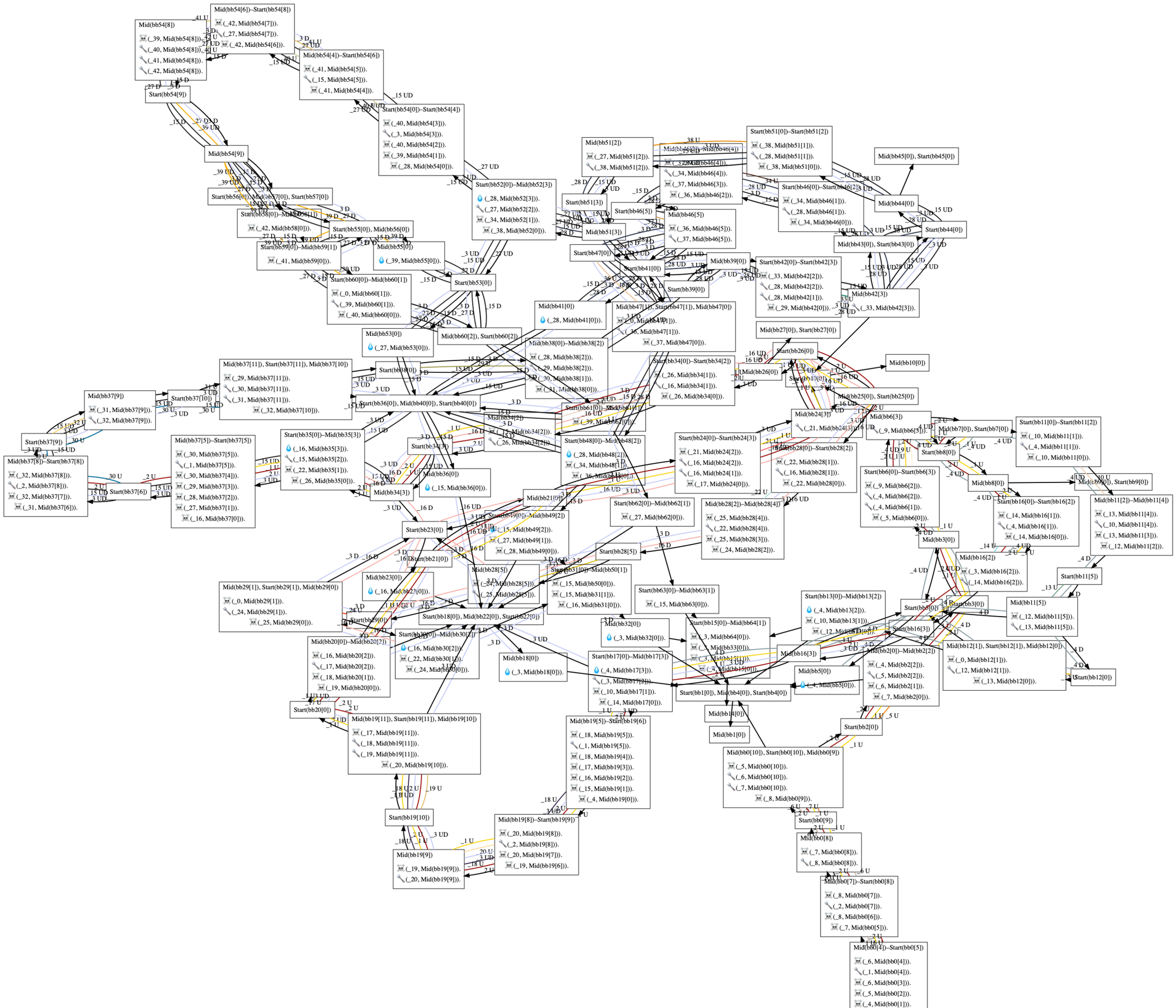


How???

```
fn foo() {
    let x = 22;
    let p = &x;
}
```

```
bb0: {
    StorageLive(_1);
    _1 = const 22i32;
    FakeRead(ForLet, _1);
    StorageLive(_2);
    _2 = &_1;
    FakeRead(ForLet, _2);
    _0 = ();
    StorageDead(_2);
    StorageDead(_1);
    return;
}
```




Mid(bb54[6])–Start(bb54[8])
(_42, Mid(bb54[7])).
(_27, Mid(bb54[7])).
(_42, Mid(bb54[6])).
Mid(bb54[8])
(_39, Mid(bb54[8])).
(_40, Mid(bb54[8])).
(_41, Mid(bb54[8])).
(_42, Mid(bb54[8])).
Mid(bb54[4])–Start(bb54[6])
(_41, Mid(bb54[5])).
(_15, Mid(bb54[5])).
(_41, Mid(bb54[4])).
Start(bb54[9])
Start(bb54[0])–Start(bb54[4])
(_40, Mid(bb54[3])).
(_3, Mid(bb54[3])).
(_40, Mid(bb54[2])).
(_39, Mid(bb54[1])).
(_28, Mid(bb54[0])).
Mid(bb54[9])
Start(bb56[0])–Mid(bb57[0]), Start(bb57[0])
Start(bb58[0])–Mid(bb58[1])
(_42, Mid(bb58[0])).
Start(bb55[0]), Mid(bb56[0])
Start(bb59[0])–Mid(bb59[1])
(_41, Mid(bb59[0])).
Mid(bb55[0])
(_39, Mid(bb55[0])).
Start(bb60[0])–Mid(bb60[1])
(_0, Mid(bb60[1])).
(_39, Mid(bb60[1])).
(_40, Mid(bb60[0])).
Start(bb52[0])–Mid(bb52[3])
(_28, Mid(bb52[3])).
(_27, Mid(bb52[2])).
(_34, Mid(bb52[1])).
(_38, Mid(bb52[0])).
Start(bb53[0])
Mid(bb53[0])
(_27, Mid(bb53[0])).
Mid(bb60[2]), Start(bb60[2])
Mid(bb51[2])
(_27, Mid(bb51[2])).
(_38, Mid(bb51[2])).
Start(bb51[3])
Mid(bb51[3])
Start(bb46[5])
Mid(bb46[5])
(_36, Mid(bb46[5])).
(_37, Mid(bb46[5])).
Start(bb47[0])
Start(bb41[0])
Start(bb39[0])
Mid(bb39[0])
Start(bb51[0])–Start(bb51[2])
(_38, Mid(bb51[1])).
(_28, Mid(bb51[1])).
(_38, Mid(bb51[0])).
Start(bb46[0])–Start(bb46[2])
(_34, Mid(bb46[1])).
(_28, Mid(bb46[1])).
(_34, Mid(bb46[0])).
Mid(bb45[0]), Start(bb45[0])
Mid(bb44[0])
Start(bb44[0])
Mid(bb43[0]), Start(bb43[0])
Start(bb42[0])–Start(bb42[3])
(_33, Mid(bb42[2])).
(_28, Mid(bb42[2])).
(_28, Mid(bb42[1])).
(_29, Mid(bb42[0])).
Mid(bb42[3])
(_33, Mid(bb42[3])).
Mid(bb41[0])
(_28, Mid(bb41[0])).
Mid(bb47[1]), Start(bb47[1]), Mid(bb47[0])
(_0, Mid(bb47[1])).
(_36, Mid(bb47[1])).
(_37, Mid(bb47[0])).
Mid(bb27[0]), Start(bb27[0])
Start(bb26[0])
Mid(bb26[0])
Mid(bb10[0])
Start(bb10[0])
Mid(bb38[0])–Mid(bb38[2])
(_28, Mid(bb38[2])).
(_29, Mid(bb38[2])).
(_30, Mid(bb38[1])).
(_31, Mid(bb38[0])).
Start(bb34[0])–Start(bb34[2])
(_26, Mid(bb34[1])).
(_16, Mid(bb34[1])).
(_26, Mid(bb34[0])).
Mid(bb37[11]), Start(bb37[11]), Mid(bb37[10])
(_29, Mid(bb37[11])).
(_30, Mid(bb37[11])).
(_31, Mid(bb37[11])).
(_32, Mid(bb37[10])).
Start(bb38[0])
Start(bb37[10])
Mid(bb37[9])
(_31, Mid(bb37[9])).
(_32, Mid(bb37[9])).
Start(bb36[0]), Mid(bb40[0]), Start(bb40[0])
Start(bb61[0])–Mid(bb61[1])
(_39, Mid(bb61[0])).
Mid(bb34[2])
(_15, Mid(bb34[2])).
(_26, Mid(bb34[2])).
Start(bb37[9])
Mid(bb37[8])–Start(bb37[8])
(_32, Mid(bb37[8])).
(_2, Mid(bb37[8])).
(_32, Mid(bb37[7])).
(_31, Mid(bb37[6])).
Start(bb37[6])
Mid(bb37[5])–Start(bb37[5])
(_30, Mid(bb37[5])).
(_1, Mid(bb37[5])).
(_30, Mid(bb37[4])).
(_29, Mid(bb37[3])).
(_28, Mid(bb37[2])).
(_27, Mid(bb37[1])).
(_16, Mid(bb37[0])).
Start(bb35[0])–Mid(bb35[3])
(_16, Mid(bb35[3])).
(_15, Mid(bb35[2])).
(_22, Mid(bb35[1])).
(_26, Mid(bb35[0])).
Start(bb34[3])
Mid(bb36[0])
(_15, Mid(bb36[0])).
Mid(bb34[3])
Start(bb48[0])–Mid(bb48[2])
(_28, Mid(bb48[2])).
(_34, Mid(bb48[1])).
(_36, Mid(bb48[0])).
Start(bb24[0])–Start(bb24[3])
(_21, Mid(bb24[2])).
(_16, Mid(bb24[2])).
(_16, Mid(bb24[1])).
(_17, Mid(bb24[0])).
Mid(bb24[3])
(_21, Mid(bb24[3])).
Mid(bb28[0])–Start(bb28[2])
(_22, Mid(bb28[1])).
(_16, Mid(bb28[1])).
(_22, Mid(bb28[0])).
Mid(bb6[3])
(_9, Mid(bb6[3])).
Mid(bb7[0]), Start(bb7[0])
Start(bb11[0])–Start(bb11[2])
(_10, Mid(bb11[1])).
(_4, Mid(bb11[1])).
(_10, Mid(bb11[0])).
Start(bb8[0])
Start(bb6[0])–Start(bb6[3])
(_9, Mid(bb6[2])).
(_4, Mid(bb6[2])).
(_4, Mid(bb6[1])).
(_5, Mid(bb6[0])).
Mid(bb8[0])
Mid(bb9[0]), Start(bb9[0])
Start(bb62[0])–Mid(bb62[1])
(_27, Mid(bb62[0])).
Mid(bb28[2])–Mid(bb28[4])
(_25, Mid(bb28[4])).
(_22, Mid(bb28[4])).
(_25, Mid(bb28[3])).
(_24, Mid(bb28[2])).
Start(bb16[0])–Start(bb16[2])
(_14, Mid(bb16[1])).
(_4, Mid(bb16[1])).
(_14, Mid(bb16[0])).
Mid(bb11[2])–Mid(bb11[4])
(_13, Mid(bb11[4])).
(_10, Mid(bb11[4])).
(_13, Mid(bb11[3])).
(_12, Mid(bb11[2])).
Start(bb11[5])
Mid(bb3[0])
Mid(bb16[2])
(_3, Mid(bb16[2])).
(_14, Mid(bb16[2])).
Start(bb49[0])–Mid(bb49[2])
(_15, Mid(bb49[2])).
(_27, Mid(bb49[1])).
(_28, Mid(bb49[0])).
Start(bb21[0])
Mid(bb21[0])
Start(bb23[0])
Mid(bb23[0])
(_16, Mid(bb23[0])).
Start(bb28[5])
Mid(bb28[5])
(_24, Mid(bb28[5])).
(_25, Mid(bb28[5])).
Start(bb31[0])–Mid(bb50[1])
(_15, Mid(bb50[0])).
(_15, Mid(bb31[1])).
(_16, Mid(bb31[0])).
Start(bb63[0])–Mid(bb63[1])
(_15, Mid(bb63[0])).
Mid(bb29[1]), Start(bb29[1]), Mid(bb29[0])
(_0, Mid(bb29[1])).
(_24, Mid(bb29[1])).
(_25, Mid(bb29[0])).
Start(bb29[0])
Start(bb18[0]), Mid(bb22[0]), Start(bb22[0])
Start(bb30[0])–Mid(bb30[2])
(_16, Mid(bb30[2])).
(_22, Mid(bb30[1])).
(_24, Mid(bb30[0])).
Mid(bb32[0])
(_3, Mid(bb32[0])).
Start(bb15[0])–Mid(bb64[1])
(_3, Mid(bb64[0])).
(_3, Mid(bb33[0])).
(_3, Mid(bb15[1])).
(_4, Mid(bb15[0])).
Mid(bb20[0])–Mid(bb20[2])
(_16, Mid(bb20[2])).
(_17, Mid(bb20[2])).
(_18, Mid(bb20[1])).
(_19, Mid(bb20[0])).
Mid(bb18[0])
(_3, Mid(bb18[0])).
Start(bb17[0])–Mid(bb17[3])
(_4, Mid(bb17[3])).
(_3, Mid(bb17[2])).
(_10, Mid(bb17[1])).
(_14, Mid(bb17[0])).
Start(bb20[0])
Mid(bb19[11]), Start(bb19[11]), Mid(bb19[10])
(_17, Mid(bb19[11])).
(_18, Mid(bb19[11])).
(_19, Mid(bb19[11])).
(_20, Mid(bb19[10])).
Start(bb19[10])
Mid(bb19[9])
(_19, Mid(bb19[9])).
(_20, Mid(bb19[9])).
Mid(bb19[8])–Start(bb19[9])
(_20, Mid(bb19[8])).
(_2, Mid(bb19[8])).
(_20, Mid(bb19[7])).
(_19, Mid(bb19[6])).
Mid(bb19[5])–Start(bb19[6])
(_18, Mid(bb19[5])).
(_1, Mid(bb19[5])).
(_18, Mid(bb19[4])).
(_17, Mid(bb19[3])).
(_16, Mid(bb19[2])).
(_15, Mid(bb19[1])).
(_4, Mid(bb19[0])).
Start(bb1[0]), Mid(bb4[0]), Start(bb4[0])
Mid(bb16[3])
Start(bb13[0])–Mid(bb13[2])
(_4, Mid(bb13[2])).
(_10, Mid(bb13[1])).
(_12, Mid(bb13[0])).
Mid(bb5[0])
(_4, Mid(bb5[0])).
Start(bb5[0])
Start(bb16[3])
Start(bb3[0])
Mid(bb2[0])–Mid(bb2[2])
(_4, Mid(bb2[2])).
(_5, Mid(bb2[2])).
(_6, Mid(bb2[1])).
(_7, Mid(bb2[0])).
Mid(bb11[5])
(_12, Mid(bb11[5])).
(_13, Mid(bb11[5])).
Mid(bb12[1]), Start(bb12[1]), Mid(bb12[0])
(_0, Mid(bb12[1])).
(_12, Mid(bb12[1])).
(_13, Mid(bb12[0])).
Start(bb12[0])
Mid(bb14[0])
Mid(bb1[0])
Start(bb2[0])
Mid(bb0[10]), Start(bb0[10]), Mid(bb0[9])
(_5, Mid(bb0[10])).
(_6, Mid(bb0[10])).
(_7, Mid(bb0[10])).
(_8, Mid(bb0[9])).
Start(bb0[9])
Mid(bb0[8])
(_7, Mid(bb0[8])).
(_8, Mid(bb0[8])).
Mid(bb0[7])–Start(bb0[8])
(_8, Mid(bb0[7])).
(_2, Mid(bb0[7])).
(_8, Mid(bb0[6])).
(_7, Mid(bb0[5])).
Mid(bb0[4])–Start(bb0[5])
(_6, Mid(bb0[4])).
(_1, Mid(bb0[4])).
(_6, Mid(bb0[3])).
(_5, Mid(bb0[2])).
(_4, Mid(bb0[1])).

What dreams may come?

What dreams may come?

- (de-)initialisation tracking (also impacts liveness)

What dreams may come?

- (de-)initialisation tracking (also impacts liveness)
- re-formulation of the subset rules upon set equality

What dreams may come?

- (de-)initialisation tracking (also impacts liveness)
- re-formulation of the subset rules upon set equality
- illegal subset relations

What dreams may come?

- (de-)initialisation tracking (also impacts liveness)
- re-formulation of the subset rules upon set equality
- illegal subset relations
- higher-ranked provenances (e.g. `f<'b>(&'b u32)`)

What dreams may come?

- (de-)initialisation tracking (also impacts liveness)
- re-formulation of the subset rules upon set equality
- illegal subset relations
- higher-ranked provenances (e.g. `f<'b>(&'b u32)`)
- validation

What dreams may come?

- (de-)initialisation tracking (also impacts liveness)
- re-formulation of the subset rules upon set equality
- illegal subset relations
- higher-ranked provenances (e.g. `f<'b>(&'b u32)`)
- validation
- pre-benchmarking

Summary

Summary

- Polonius: a reformulation of the Rust borrow check

Summary

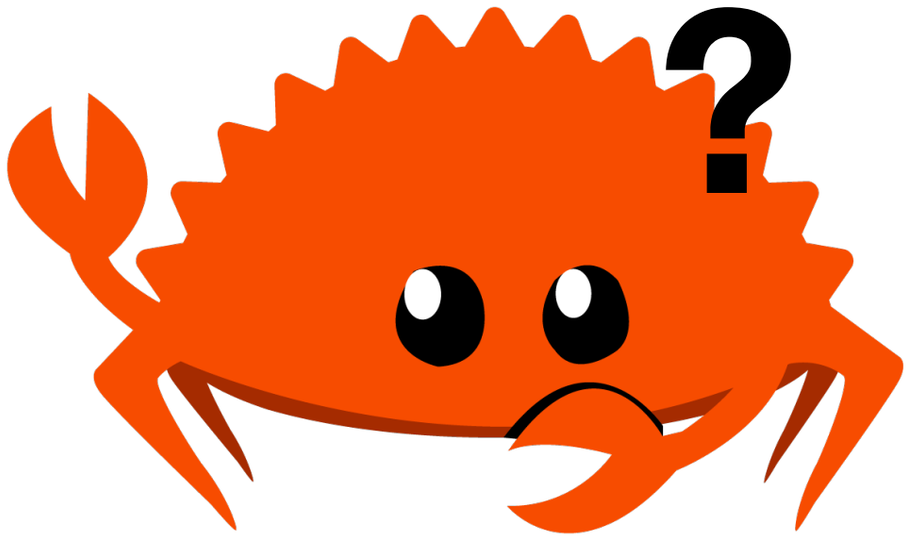
- Polonius: a reformulation of the Rust borrow check
- Provenance variables in types = sets of possible source loans

Summary

- Polonius: a reformulation of the Rust borrow check
- Provenance variables in types = sets of possible source loans
- Declaratively defined in “Datalog”

Summary

- Polonius: a reformulation of the Rust borrow check
- Provenance variables in types = sets of possible source loans
- Declaratively defined in “Datalog”
- Example of prov-var liveness

Summary

- Polonius: a reformulation of the Rust borrow check
- Provenance variables in types = sets of possible source loans
- Declaratively defined in “Datalog”
- Example of prov-var liveness