Clang Static Analyzer: Supporting Multithreaded Code

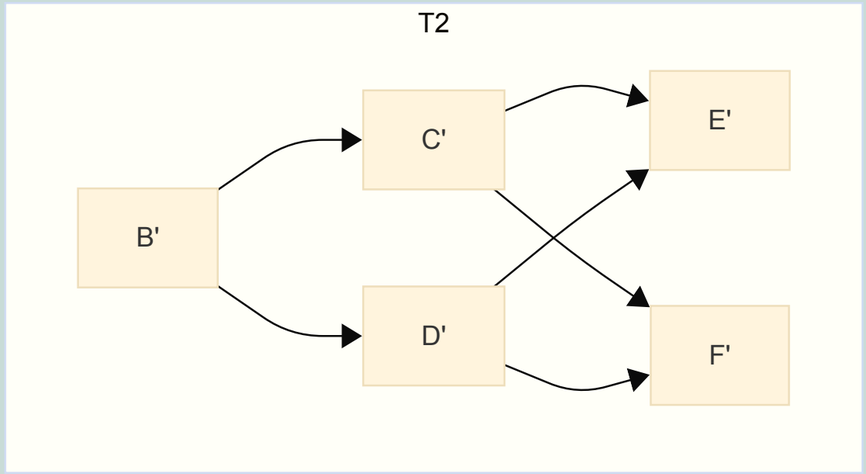
Isaac Nudelman

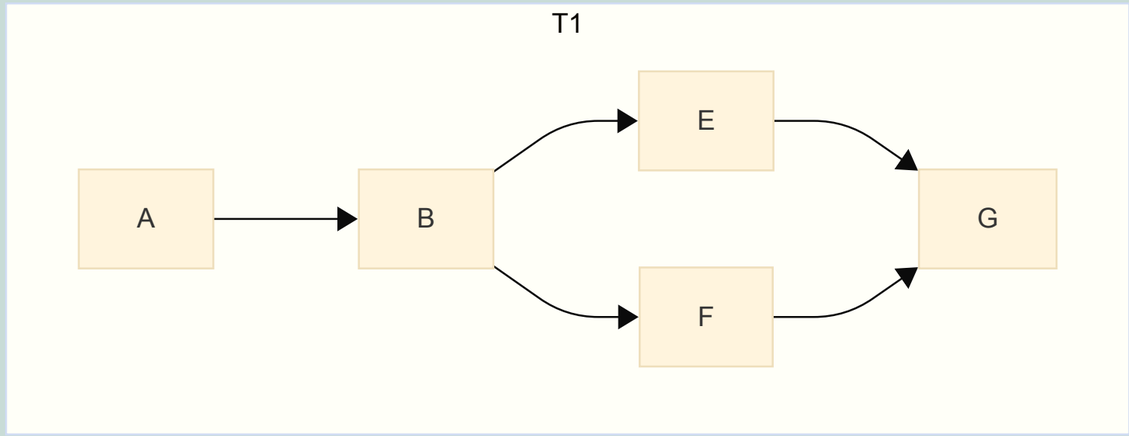
Spot the bug

```
void *thread(void *arg)
{
    auto *res = (Resource *)arg;
    use_resource(res);
    // ...
}

void main()
{
    pthread_t tid;
    Resource my_resource;
    pthread_create(&tid, NULL, thread, &my_resource);
    init_resource(&my_resource);
    // ...
    pthread_join(tid, nullptr);
}
```

Threads become islands of state





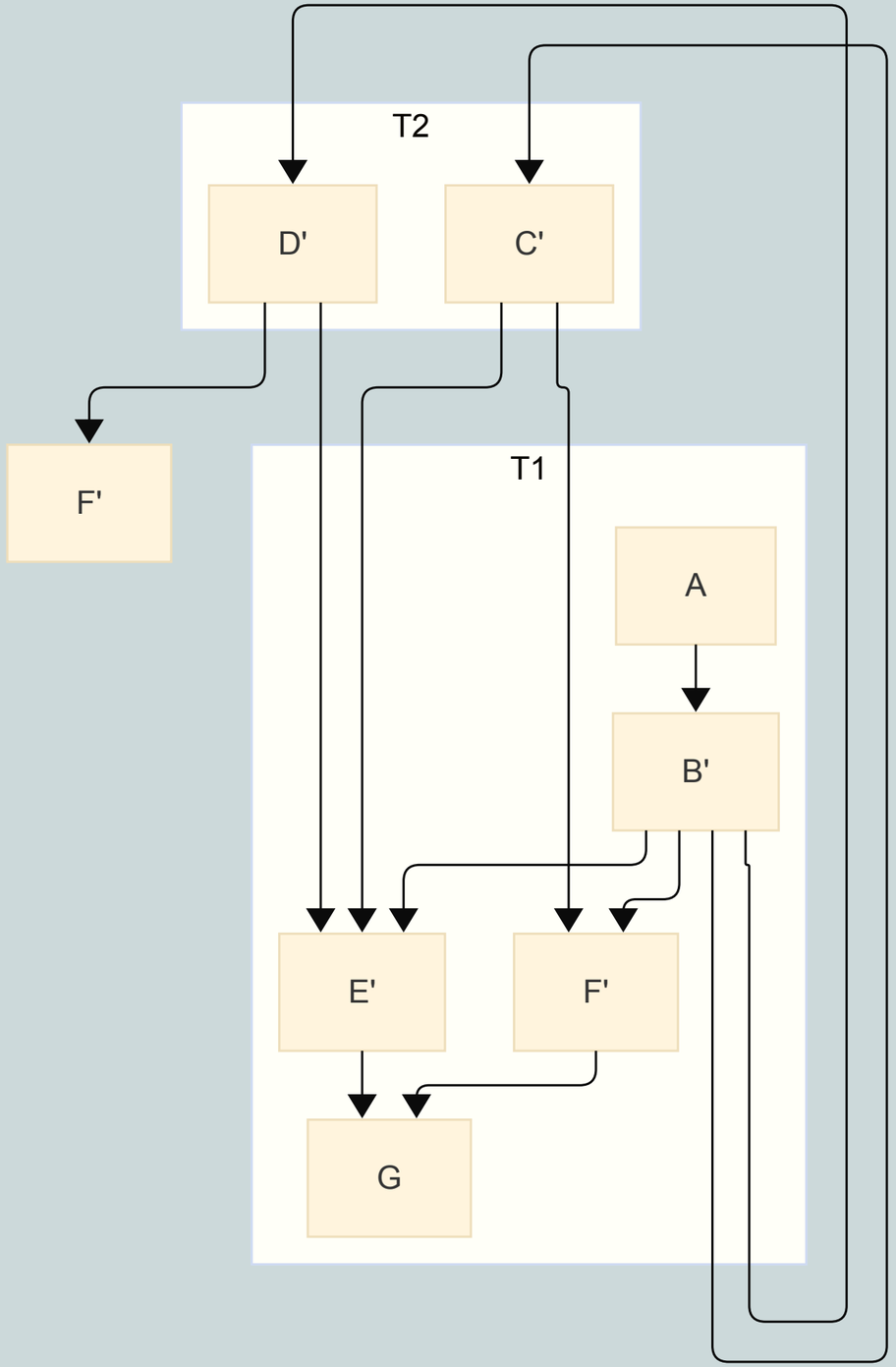
Now:

1. bridge the islands

Eventually:

Data Races

Deadlocks



Goals

1. checkers just work™
2. `sizeof(CHAR_BIT) == 42` OK

Not Supported

SIMT (CUDA)

All threads look the same to the analyzer

```
thread_create(..., fn, data);
```

↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓

```
(void)fn; (void)data;
```

Need to bridge state islands directly

Inlining

> When all you have is a hammer,
> everything becomes a nail

The first attempt

Success on the second first try

```
thread_create(fn,
data);
```

```
fn(data);
```

Validation

```
void* thread(void* arg)
{
    clang_analyzer_checkInlined(true);// expected-warning{{TRUE}}
    return NULL;
}

int ok()
{
    pthread_t p1;
    pthread_create(&p1, NULL, &thread_function, NULL);
    return 0;
}
```

Results

1. existing checkers just work™
2. negligible overhead[1]
3. can find bugs in real code

1. caveat: didn't have time for proper characterization

Future?

1. Upstream
2. Start writing multithreading checks
3. Iterate?

Thank You!

Q&A

isaac.nudelman@utexas.edu

in/isaac-nudelman