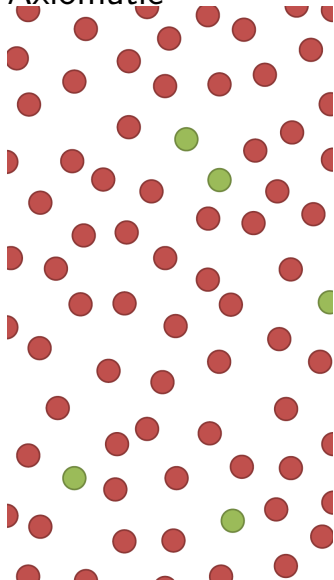


An operational semantics of C11 concurrency

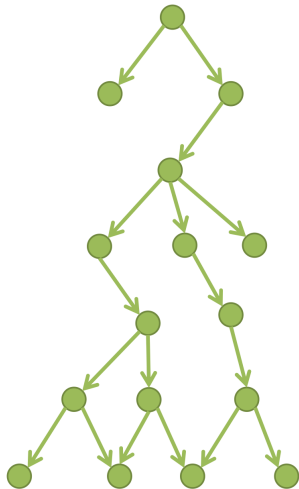
Kyndylan Nienhuis

September 24, 2014

Axiomatic



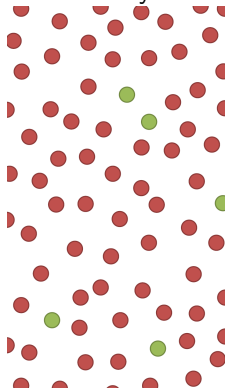
Operational



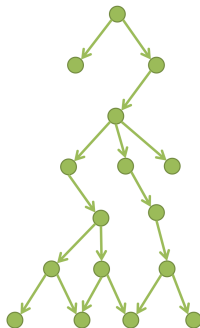
Why do we want an operational semantics of concurrency?

The C11 standard uses both styles of semantics:

An axiomatic style for
concurrency

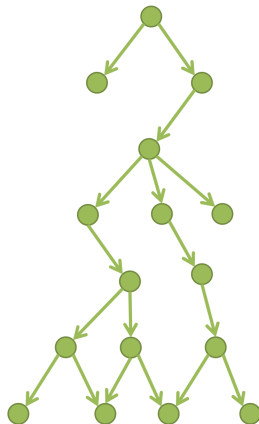
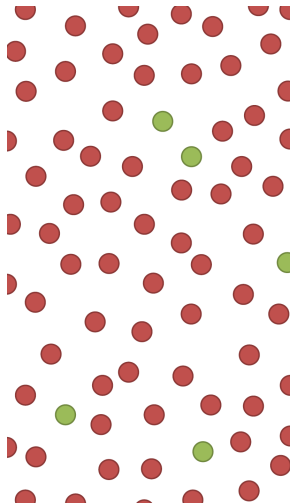


and an operational style for
the rest

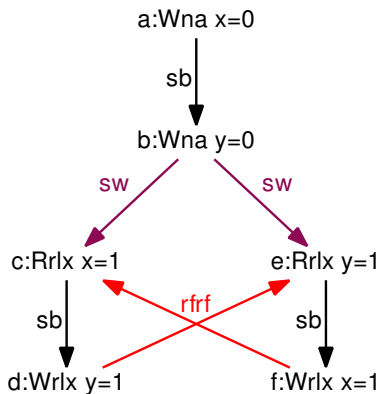


Why do we want an operational semantics of concurrency?

So we can compute (some) behaviour of large or infinite programs.

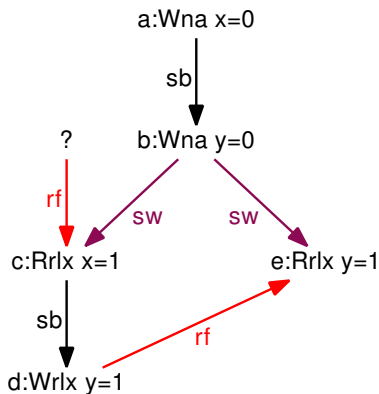


Generating relaxed behaviour



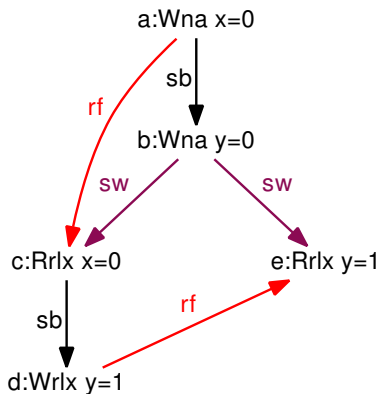
```
atomic_int x=0;
atomic_int y=0;
{{{
r1 = x.load(mo_relaxed);
y.store(1, mo_relaxed);
|||
r2 = y.load(mo_relaxed);
x.store(1, mo_relaxed);
}}}
```

Generating relaxed behaviour



```
atomic_int x=0;
atomic_int y=0;
{{{
r1 = x.load(mo_relaxed);
y.store(1, mo_relaxed);
}}}
r2 = y.load(mo_relaxed);
x.store(1, mo_relaxed);
}}}
```

Generating relaxed behaviour



```
atomic_int x=0;
atomic_int y=0;
{{{
r1 = x.load(mo_relaxed);
y.store(1, mo_relaxed);
|||
r2 = y.load(mo_relaxed);
x.store(1, mo_relaxed);
}}}
```

Generating relaxed behaviour

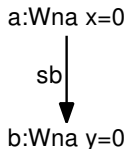
a:Wna $x=0$

The model has two parts:

- ▶ A threadlocal semantics that determines the the nodes, sb and asw
- ▶ A concurrency model that determines rf

They can take steps independently of each other. To allow this, the threadlocal semantics is symbolic.

Generating relaxed behaviour

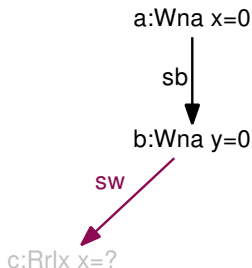


The model has two parts:

- ▶ A threadlocal semantics that determines the the nodes, sb and asw
- ▶ A concurrency model that determines rf

They can take steps independently of each other. To allow this, the threadlocal semantics is symbolic.

Generating relaxed behaviour

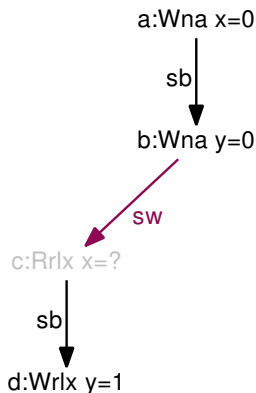


The model has two parts:

- ▶ A threadlocal semantics that determines the the nodes, sb and asw
- ▶ A concurrency model that determines rf

They can take steps independently of each other. To allow this, the threadlocal semantics is symbolic.

Generating relaxed behaviour

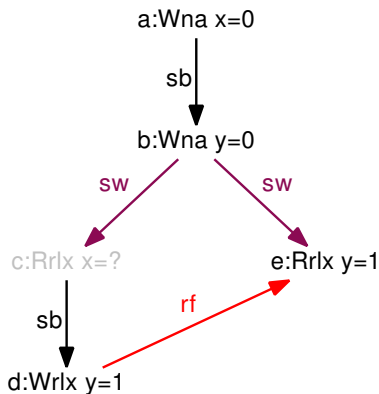


The model has two parts:

- ▶ A threadlocal semantics that determines the the nodes, `sb` and `asw`
- ▶ A concurrency model that determines `rf`

They can take steps independently of each other. To allow this, the threadlocal semantics is symbolic.

Generating relaxed behaviour

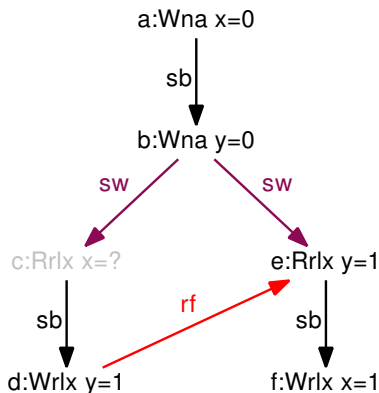


The model has two parts:

- ▶ A threadlocal semantics that determines the the nodes, sb and asw
- ▶ A concurrency model that determines rf

They can take steps independently of each other. To allow this, the threadlocal semantics is symbolic.

Generating relaxed behaviour

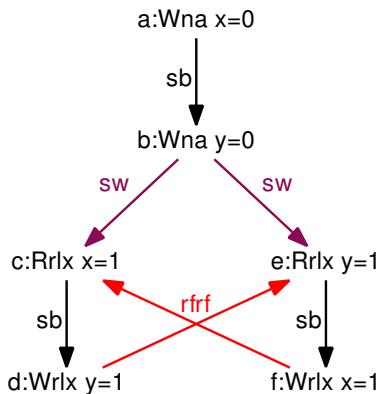


The model has two parts:

- ▶ A threadlocal semantics that determines the the nodes, sb and asw
- ▶ A concurrency model that determines rf

They can take steps independently of each other. To allow this, the threadlocal semantics is symbolic.

Generating relaxed behaviour

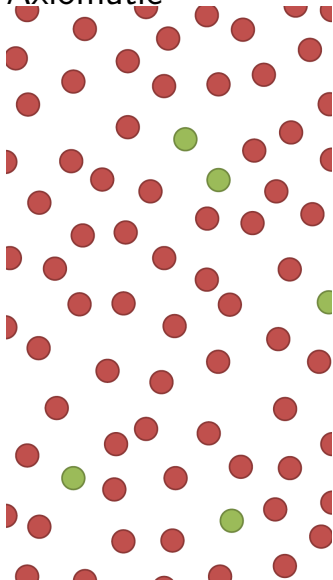


The model has two parts:

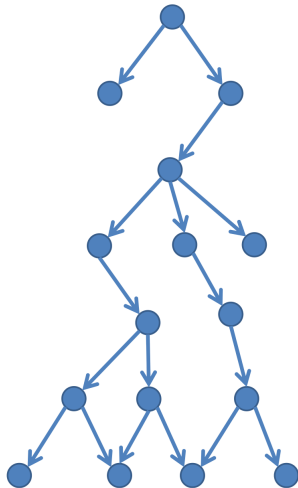
- ▶ A threadlocal semantics that determines the the nodes, sb and asw
- ▶ A concurrency model that determines rf

They can take steps independently of each other. To allow this, the threadlocal semantics is symbolic.

Axiomatic



Operational

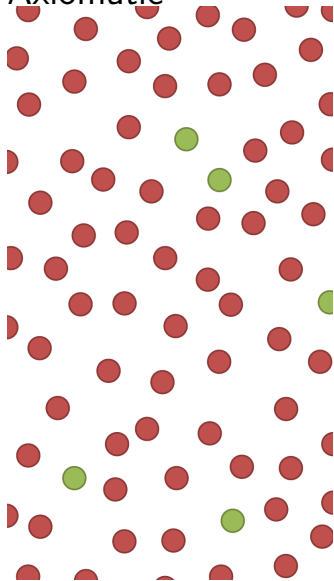


The graph consists of 15 nodes and 18 directed edges. The structure is as follows:

- Level 1 (Root):** 1 node.
- Level 2:** 2 nodes (connected from the root).
- Level 3:** 3 nodes (connected from Level 2).
- Level 4:** 5 nodes (connected from Level 3).
- Level 5 (Leaves):** 5 nodes (connected from Level 4).

The edges are directed downwards, indicating a flow from the root to the leaves.

Axiomatic



Operational

