Armada

Proof for concur. code.

"Hard concur":
- Lock-free code.
- x86-TSO.

State-machine reasoning.

Proof strategies.
- Basics: invariants, abstraction.
- Big: reduction / movers.
- Helpers: TSO elim., weakening, ...
- Levels.

Automation.

Admin.

Lab 4. published.

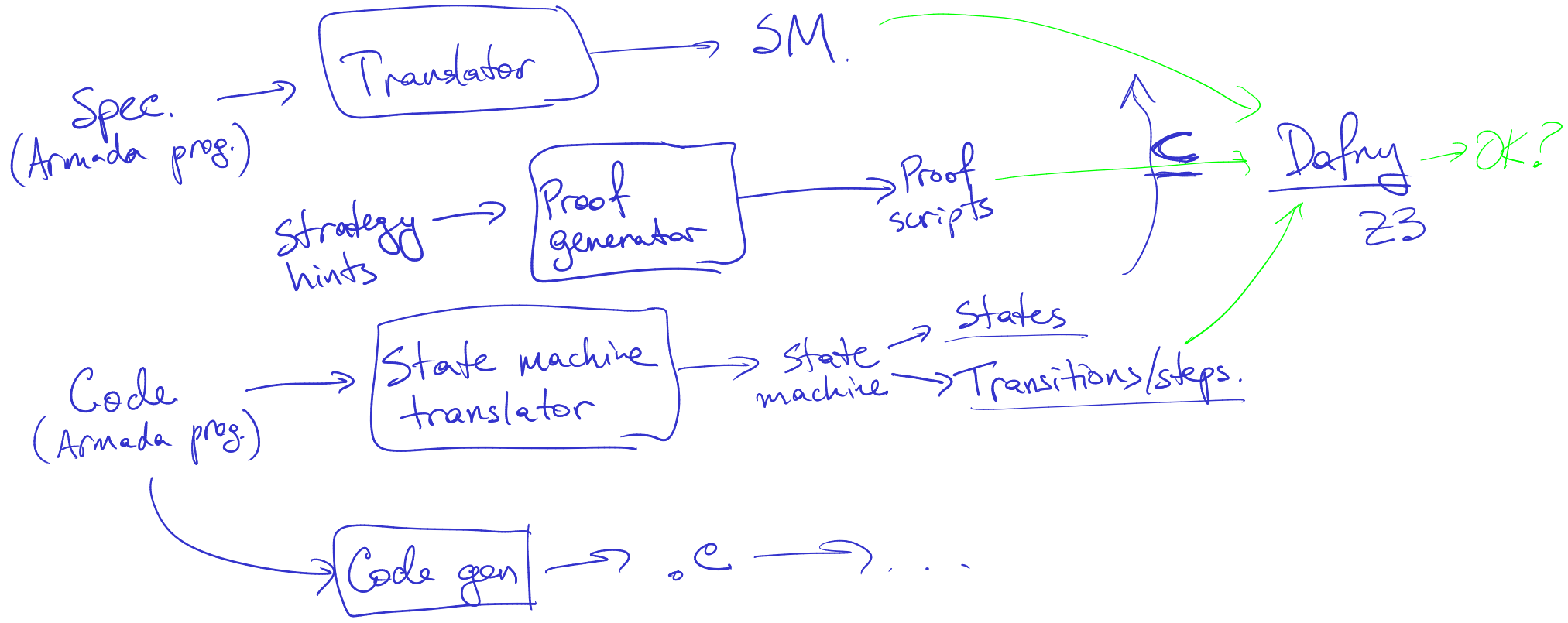
Default: replicated dbk.

Option: learn other system.

↳ Tell us!

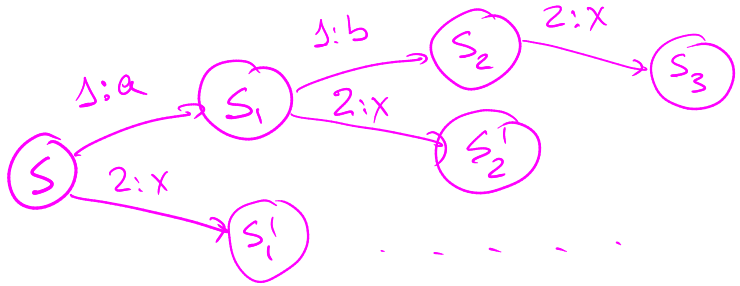
Overview

Armada language
C + spec features.



State machine concurrency

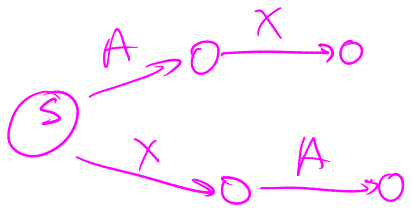
T1	T2
a	x
b	y
c	z



$\subseteq$

Spec.

T1	T2
A	X

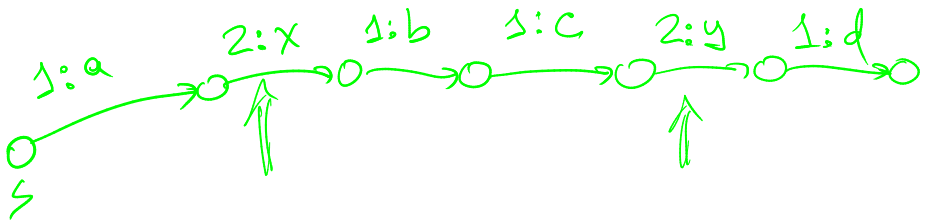


Low level

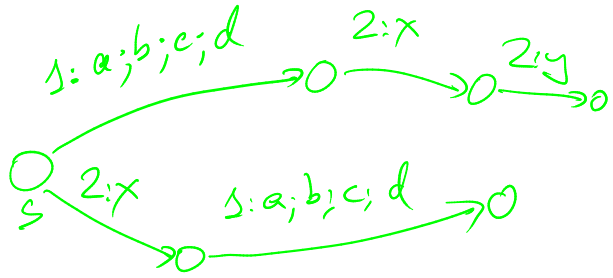
```
a  lock(&mutex);
b  while (condition()) {
c    do_something();
d    unlock(&mutex);

       lock(&mutex);
     }
     unlock(&mutex);
```

High level

```
explicit_yield {
a  lock(&mutex);
b  while (condition()) {
c    do_something();
d    unlock(&mutex);
     yield;
       lock(&mutex);
   }
   unlock(&mutex);
}
```



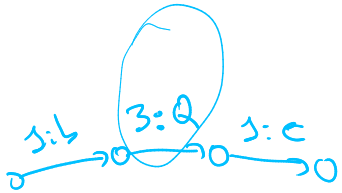
$\subseteq$

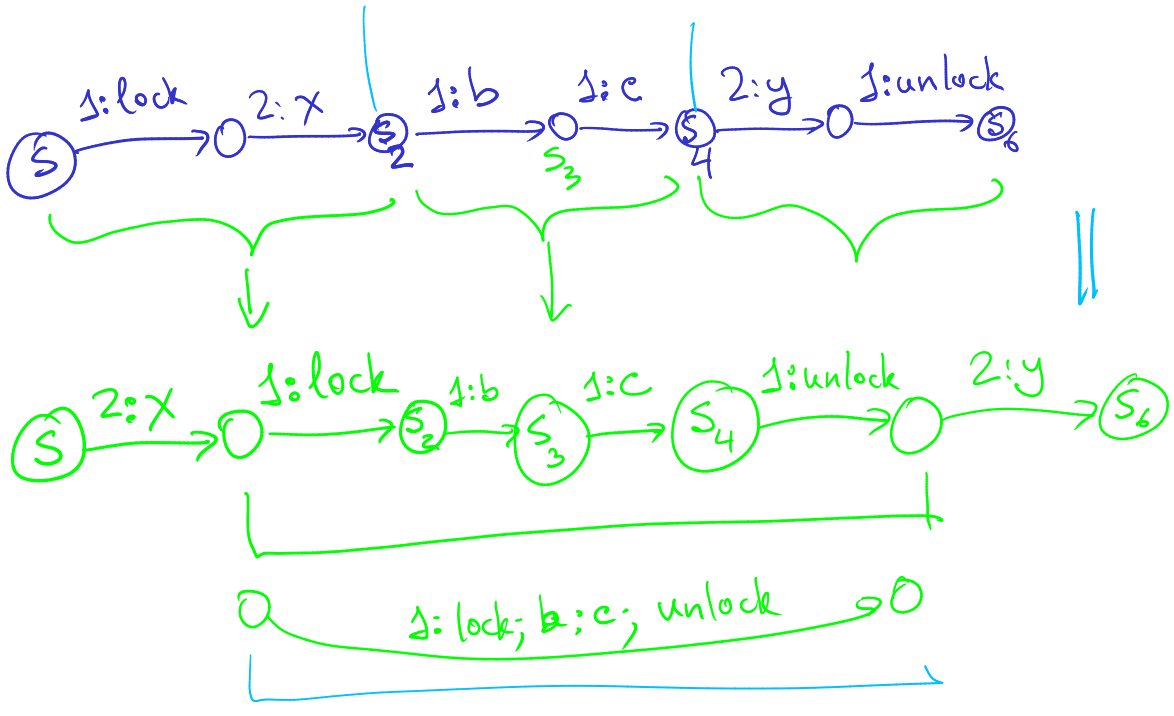


Reduction / movers

Right-movers: lock(); b; c

Left-movers: unlock(); c; b





$\subseteq$ spec

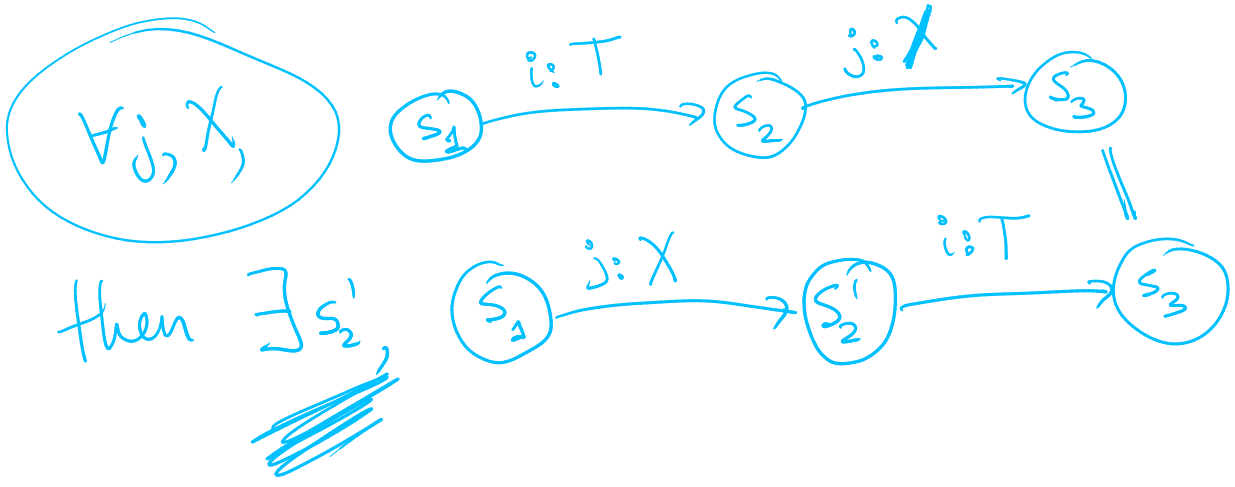
$\subseteq$ spec

General pattern: R; R; ..; R; N; L; L; ...; L.

R -- RRRRNLLLL--L

Mover proofs.

Is T a right-mover?



Leverage invariant.

i runs T

↓

i holds lock

↓

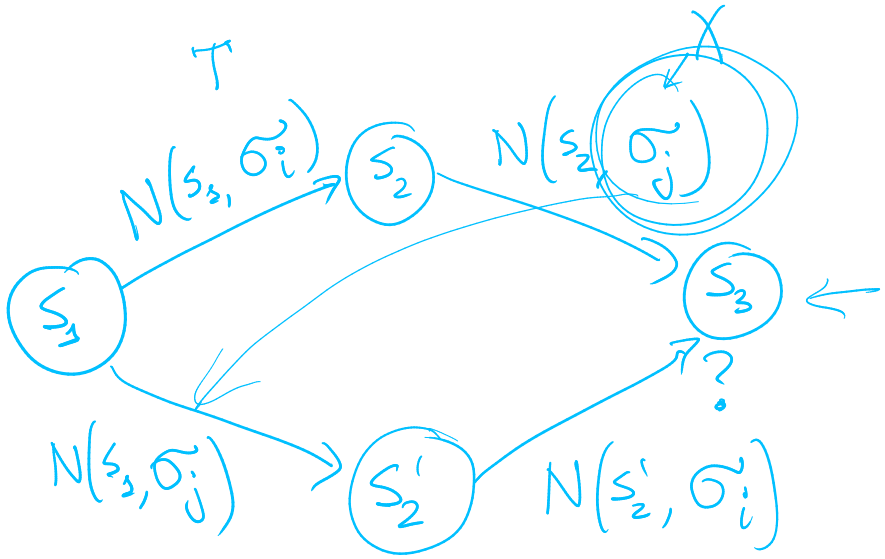
j doesn't hold lock

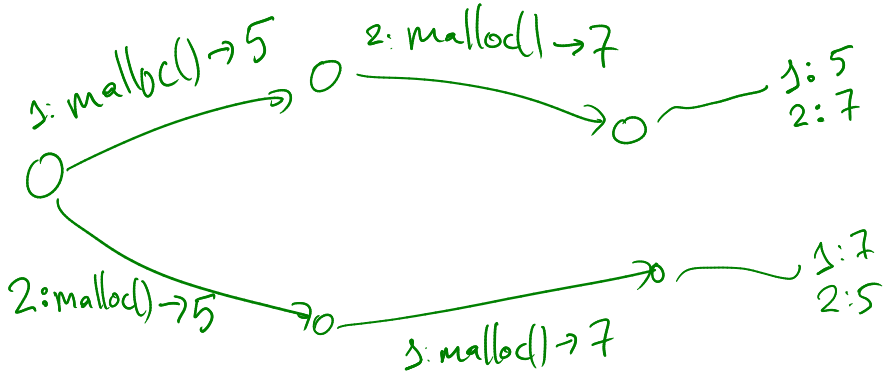
↓

j cannot run any Z if Z needs lock.

Nondet. encapsulation

NextState (state s, step σ) → state s'.



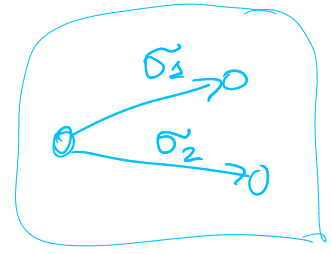


Q: what's in σ?

Thread ID.

✓ Uninitialized mem.

Branch if (*).



Runtime choices:

malloc() → ptr. 5, 7, ...

thread_create() → new tid.