Relaxations allow trading

correctness
for
performance

provide the potential
for better-performing
implementations

Relaxing the Semantics

Quantitative relaxations
Henzinger, Kirsch, Payer, Sezgin,S. POPL13

- Sequential specification = set of legal sequences
- Consistency condition = e.g. linearizability / sequential consistency

Local linearizability
Haas, Henzinger, Holzer,…, S, Veith CONCUR16

Relaxing the Sequential Specification

Quantitative relaxations (POPL13)

Goal

Stack - incorrect behavior

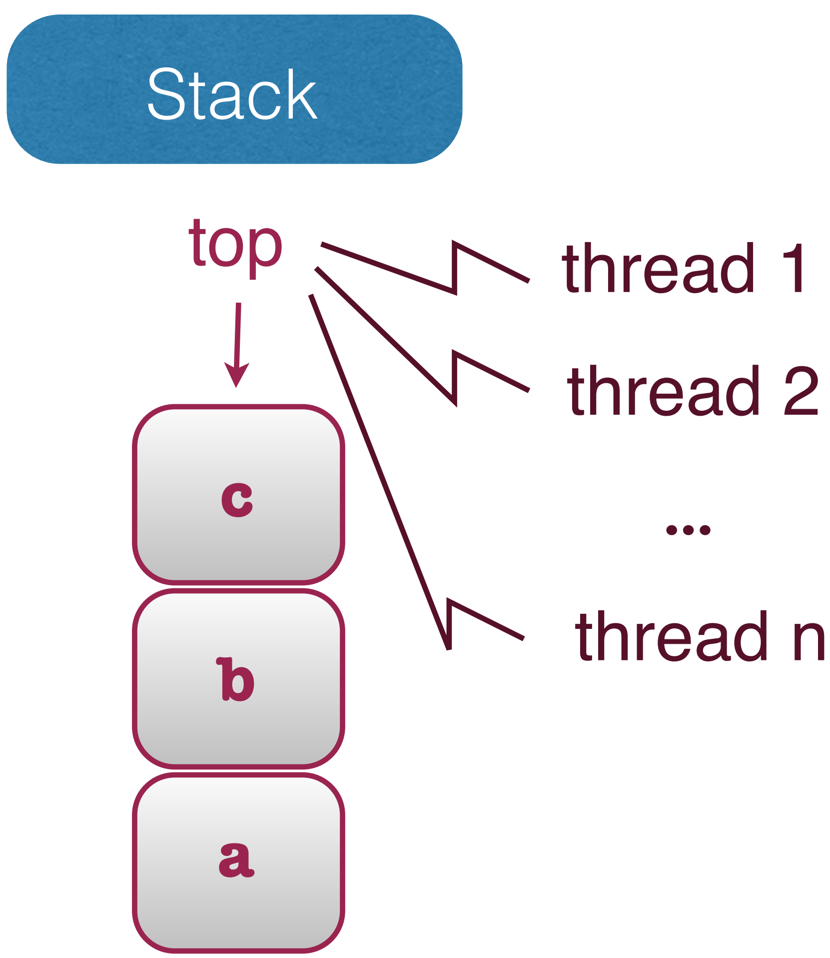
push(a)push(b)push(c)pop(a)pop(b)

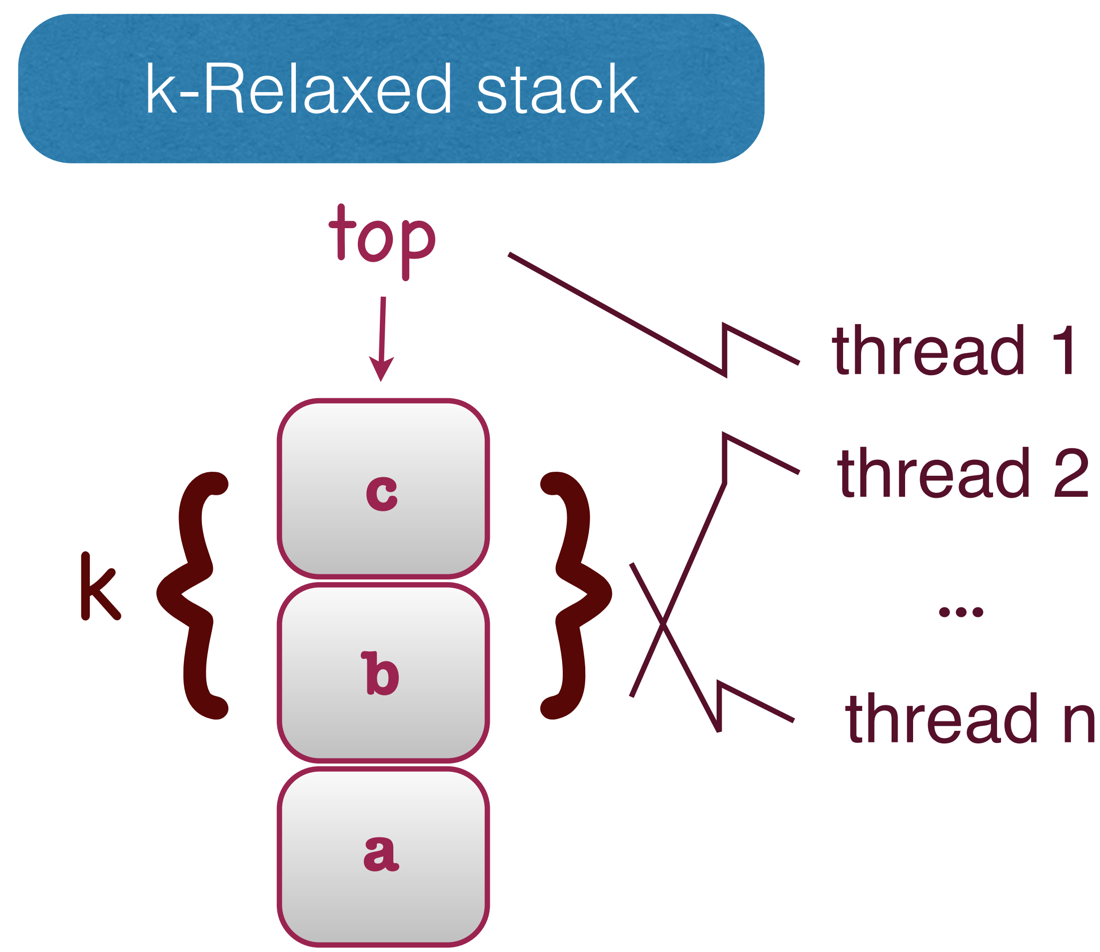
- trade correctness for performance
- in a controlled way with quantitative bounds

measure the error from correct behaviour

correct in a relaxed stack
... 2-relaxed? 3-relaxed?

How can relaxing help?





What we have

- Framework
- Generic examples
- Concrete relaxation examples
- Efficient concurrent implementations

for semantic relaxations

out-of-order / stuttering

stacks, queues, priority queues,.. / CAS, shared counter

of relaxation instances

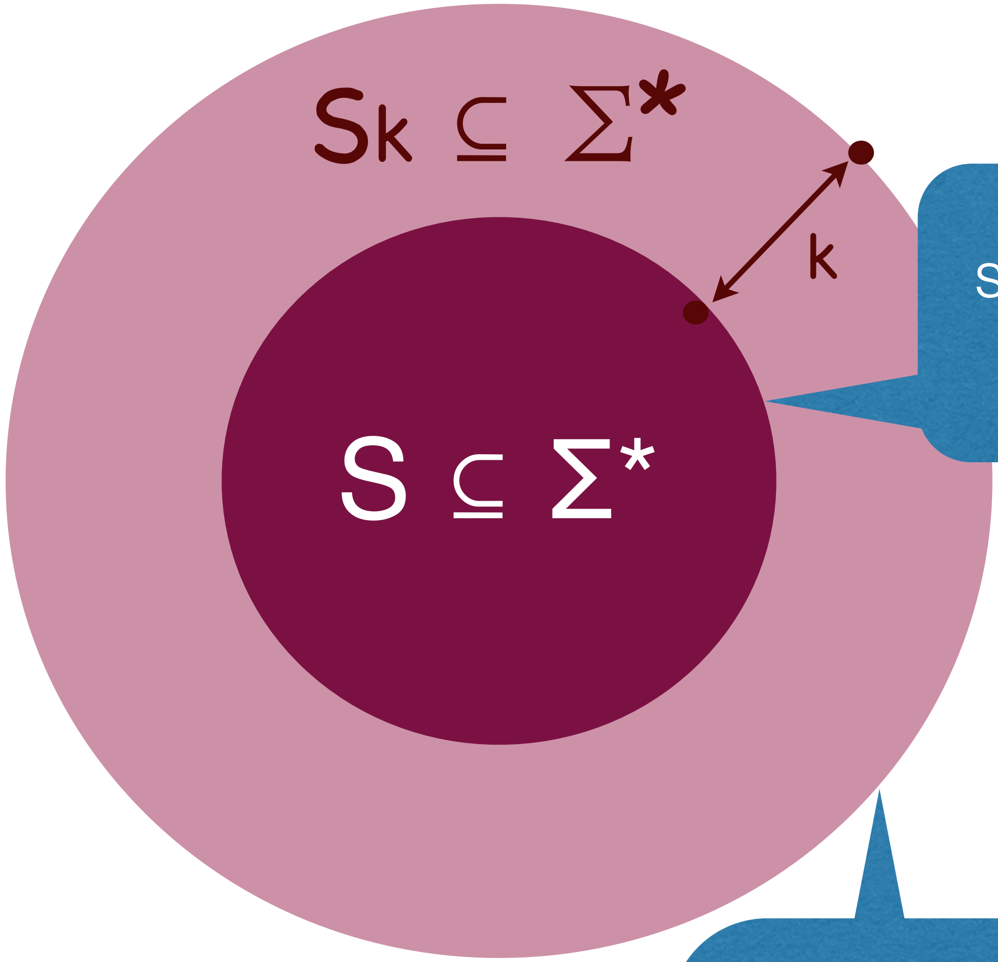
The big picture

$S \subseteq \Sigma^*$

sequential specification
legal sequences

Σ - methods with arguments

The big picture



sequential specification
legal sequences

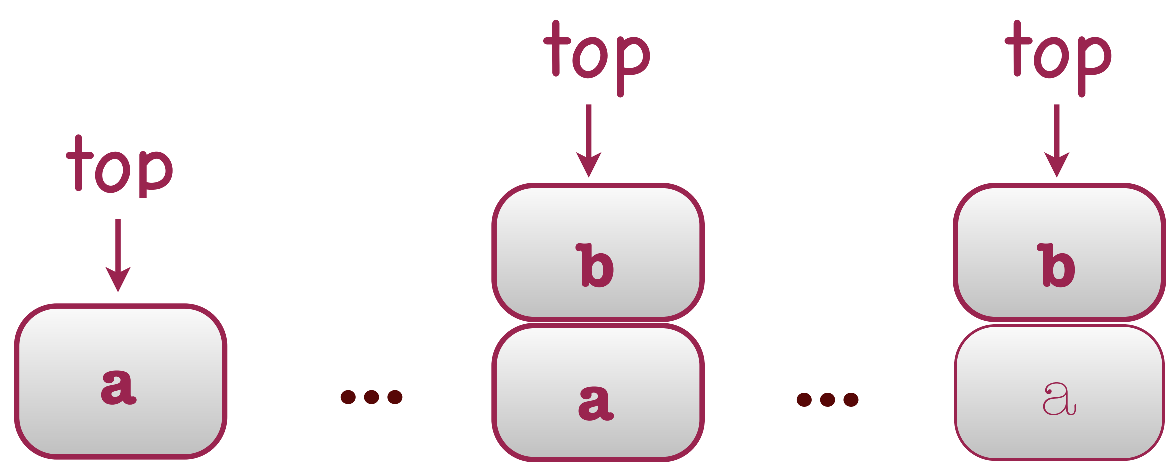
relaxed sequential specification
sequences at distance up to k from S

Σ - methods with arguments

Syntactic distances do not help

$\text{push}(a)[\text{push}(i)\text{pop}(i)]^n\text{push}(b)[\text{push}(j)\text{pop}(j)]^m\text{pop}(a)$

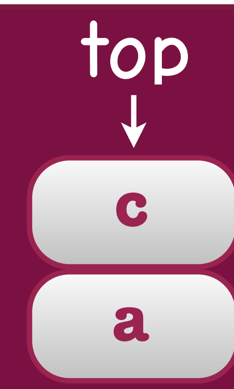
is a 1-out-of-order stack sequence



its permutation distance is min(2n,2m)

Semantic distances need a notion of state

state



- States are equivalence classes of sequences in S

example: for stack

$$\mathrm{push(a)push(b)pop(b)push(c)} \equiv \mathrm{push(a)push(c)}$$

- Two sequences in S are equivalent iff they have an indistinguishable future

$$x \equiv y \quad \Leftrightarrow \quad \forall u \in \Sigma^*.\,(xu \in S \Leftrightarrow yu \in S)$$

Semantics goes operational

$S \subseteq \Sigma^*$ is the sequential specification

states

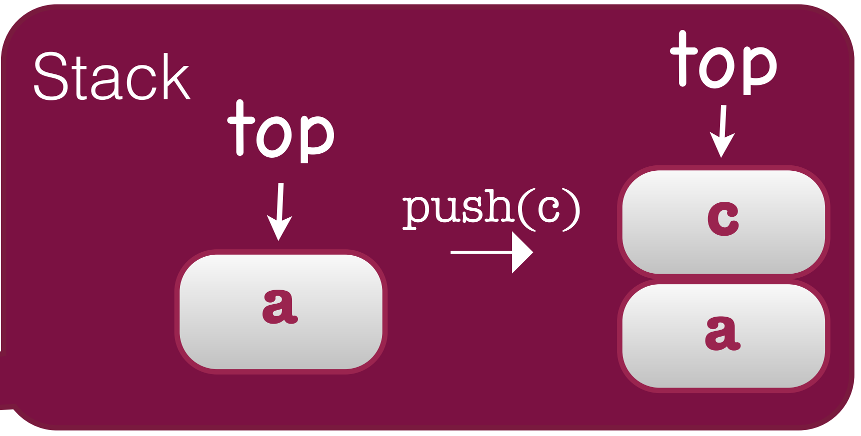
labels

initial state

$LTS(S) = (S/_{\equiv}, \Sigma, \rightarrow, [\varepsilon]_{\equiv})$ with

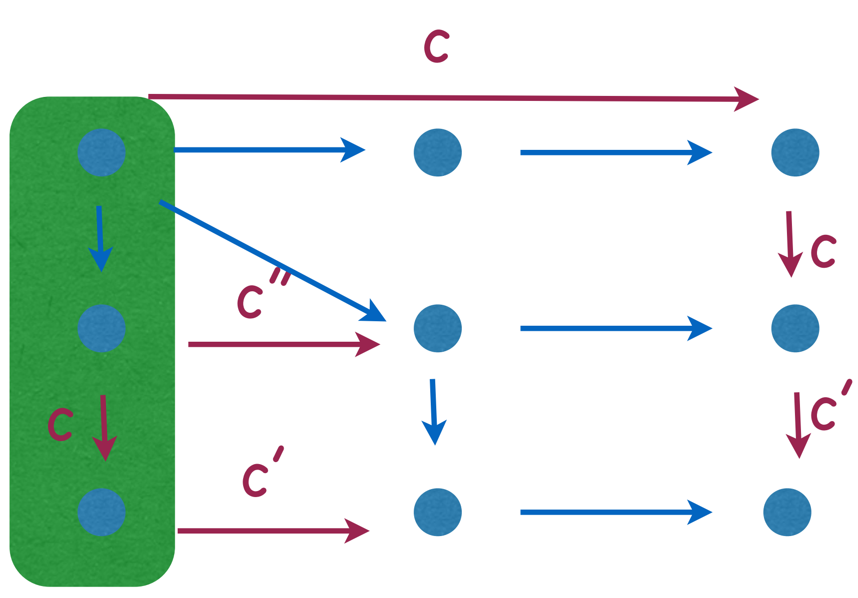
transition relation

$$[s]_{\equiv} \xrightarrow{m} [sm]_{\equiv} \quad \Leftrightarrow \quad sm \in S$$



The relaxation framework

- Start from LTS(S)
- Add transitions with transition costs
- Fix a path cost function



distance = minimal cost on all paths labelled by the sequence

Generic out-of-order

$\texttt{segment\_cost}(\,q \xrightarrow{m} q'\,) = |\mathbf{v}|$

transition cost

Where **v** is a sequence of minimal length s.t.

removing **v** enables a transition

or

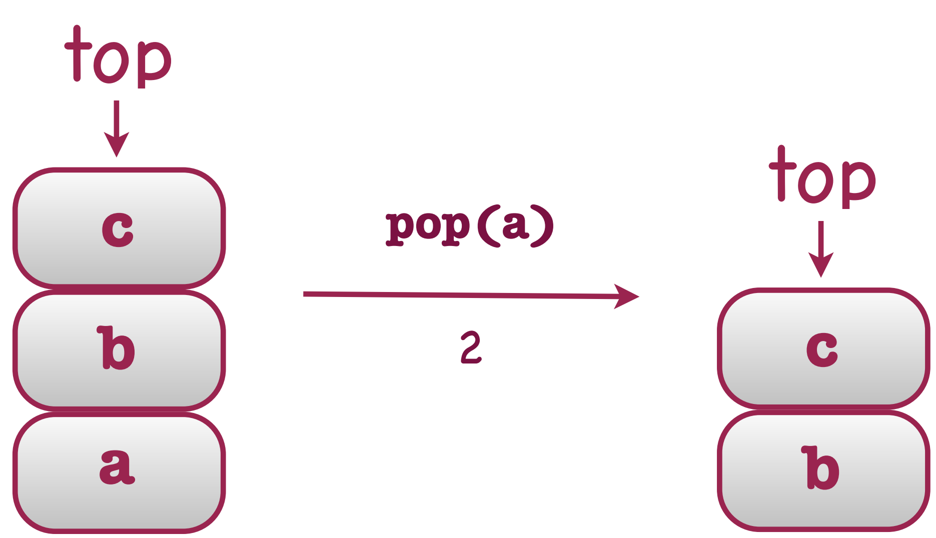
inserting **v** enables a transition

goes with different path costs

Out-of-order stack

Sequence of **push**'s with no matching **pop**

- Canonical representative of a state
- Add incorrect transitions with segment-costs



- Possible path cost functions max, sum,...

also more advanced