

Explain lock-free stack ABA problem to first-year CS student: A Visual Walkthrough (intuition-to-formalism, clean-whiteboard)

Intuition: The Simple View

Ada: "I thought Explain lock-free stack ABA problem to first-year CS student was simpler."
Turing: Let's unpack Explain lock-free stack ABA problem...

Formal-Rule: Introducing Versions

Turing: Exactly. That's the ABA problem. The value "A" is back, but it's a new node. The simple check fails.

Confusion: The "A-B-A" Scenario

Ada: "Wait, it looks the same, but it's different! How does the lock-free code know?"
Turing: Exactly. That's the ABA problem. The value "A" is back, but it's a new node. The simple check fails.

FORMAL RULE: Use a Version Counter. Every time a node is popped, increment a global counter and tag the node. Pushing a node also uses the current counter.

Ada: "Oh! So the version number acts like a unique timestamp."
Turing: Precisely. The formal rule distinguishes between the old "A" and the new "A".

Recap: Intuition to Formalism Mapping

Ada: "The version counter makes the "same" value distinct."
Turing: Yes. We translated the intuitive confusion into a formal, technical solution.

Panel 1: Teach: Explain lock-free stack ABA problem to first-year CS student

Ada: "I thought Explain lock-free stack ABA problem to first-year CS student was simpler."
Turing: Let's unpack Explain lock-free stack ABA problem...

Panel 2: Teach: Explain lock-free stack ABA problem to first-year CS student

Intuition: The Simple View

Ada: "I thought Explain lock-free stack ABA problem to first-year CS student was simpler."
Turing: Let's unpack Explain lock-free stack ABA problem...

Formal-Rule: Introducing Versions

Turing: Exactly. That's the ABA problem. The value "A" is back, but it's a new node. The simple check fails.

Confusion: The "A-B-A" Scenario

Ada: "Wait, it looks the same, but it's different! How does the lock-free code know?"
Turing: Exactly. That's the ABA problem. The value "A" is back, but it's a new node. The simple check fails.

FORMAL RULE: Use a Version Counter. Every time a node is popped, increment a global counter and tag the node. Pushing a node also uses the current counter.

Ada: "Oh! So the version number acts like a unique timestamp."
Turing: Precisely. The formal rule distinguishes between the old "A" and the new "A".

Recap: Intuition to Formalism Mapping

Ada: "The version counter makes the "same" value distinct."
Turing: Yes. We translated the intuitive confusion into a formal, technical solution.

Panel 2: Teach: Explain lock-free stack ABA problem to first-year CS student

Ada: "I thought Explain lock-free stack ABA problem to first-year CS student was simpler."
Turing: Let's unpack Explain lock-free stack ABA problem...

Panel 3: Teach: Explain lock-free stack ABA problem to first-year CS student

Intuition: The Simple View

Ada: "I thought Explain lock-free stack ABA problem to first-year CS student was simpler."
Turing: Let's unpack Explain lock-free stack ABA problem...

Formal-Rule: Introducing Versions

Turing: Exactly. That's the ABA problem. The value "A" is back, but it's a new node. The simple check fails.

Confusion: The "A-B-A" Scenario

Ada: "Wait, it looks the same, but it's different! How does the lock-free code know?"
Turing: Exactly. That's the ABA problem. The value "A" is back, but it's a new node. The simple check fails.

FORMAL RULE: Use a Version Counter. Every time a node is popped, increment a global counter and tag the node. Pushing a node also uses the current counter.

Ada: "Oh! So the version number acts like a unique timestamp."
Turing: Precisely. The formal rule distinguishes between the old "A" and the new "A".

Recap: Intuition to Formalism Mapping

Ada: "The version counter makes the "same" value distinct."
Turing: Yes. We translated the intuitive confusion into a formal, technical solution.

Panel 3: Teach: Explain lock-free stack ABA problem to first-year CS student

Ada: "I thought Explain lock-free stack ABA problem to first-year CS student was simpler."
Turing: Let's unpack Explain lock-free stack ABA problem...

Panel 4: Teach: Explain lock-free stack ABA problem to first-year CS student

Intuition: The Simple View

Ada: "I thought Explain lock-free stack ABA problem to first-year CS student was simpler."
Turing: Let's unpack Explain lock-free stack ABA problem...

Formal-Rule: Introducing Versions

Turing: Exactly. That's the ABA problem. The value "A" is back, but it's a new node. The simple check fails.

Confusion: The "A-B-A" Scenario

Ada: "Wait, it looks the same, but it's different! How does the lock-free code know?"
Turing: Exactly. That's the ABA problem. The value "A" is back, but it's a new node. The simple check fails.

FORMAL RULE: Use a Version Counter. Every time a node is popped, increment a global counter and tag the node. Pushing a node also uses the current counter.

Ada: "Oh! So the version number acts like a unique timestamp."
Turing: Precisely. The formal rule distinguishes between the old "A" and the new "A".

Recap: Intuition to Formalism Mapping

Ada: "The version counter makes the "same" value distinct."
Turing: Yes. We translated the intuitive confusion into a formal, technical solution.

Panel 4: Teach: Explain lock-free stack ABA problem to first-year CS student

Ada: "I thought Explain lock-free stack ABA problem to first-year CS student was simpler."
Turing: Let's unpack Explain lock-free stack ABA problem...