EECS3342 System Specification and Refinement
Fall 2026

Digest of Lecture 11, October 22

Bridge Controller: BAPs, Sequents, and Invariant Preservation

Jackie Wang

How to use this Digest

Use this digest as a **roadmap and checklist**. It highlights the main ideas, important examples, common pitfalls, practice targets, and where to return in the course materials. It does **not** replace attending or reviewing the session, using the recording when needed, studying Jackie's annotated iPad notes and other official course materials, completing assigned work, or practising independently. Use it to identify what you understand, what you need to revisit, and what you should be able to do on your own.

If time is limited, start with the sections most relevant to your current review and return to the rest later; selective use is better than skipping the digest because it looks long.

This is an AI-assisted study roadmap reviewed by Jackie and grounded in the session transcript and Jackie's annotated iPad notes. Like any course document, it may still contain an error or omission. If something seems technically doubtful or inconsistent with the original course materials, return to those materials and ask Jackie for clarification rather than treating the digest as final authority.

Table of Contents

Main Ideas 2

Roadmap 2

Key Terms, Notation, and Structures 5

Key Examples and Demonstrations 6

Common Pitfalls and Misconceptions 7

What You Should Be Able to Do 8

Suggested Practice 9

Where to Review 9

Closing Reminder 10

Main Ideas

- **State safety and event enabledness are separate questions.** A state is safe when it satisfies the invariants; an event is enabled when its *guard* evaluates to true in the current pre-state. A conservative guard may deliberately make some safe states unreachable.
- **Event-B actions must be read as state-transition constraints, not as sequential Java assignments.** The action $n := n + 1$ is converted into the before-after predicate (BAP) $n' = n + 1$, where n is the pre-state value and n' is the post-state value. Constants do not receive primes.
- **Parallel action semantics matter.** The familiar three-line imperative swap fails as an Event-B specification because all right-hand sides refer to the same pre-state. The correct formal swap is expressed directly by $x' = y \land y' = x$.
- **Invariant preservation becomes a proof obligation.** The desired claim is that the invariants hold after an enabled event whenever the axioms, pre-state invariants, and the event guard hold before the transition.
- **Sequents provide the proof language.** A sequent separates hypotheses from a goal using the turnstile $\vdash$. The lecture introduced its syntax and implication-based meaning, including the important convention that an empty hypothesis set behaves as `true`, not `false`.
- **The first PO/VC construction used `ML_out`.** Jackie built the invariant-preservation sequent from the model's axiom, pre-state invariants, and guard, wrote the post-state invariants with primes, and then eliminated the primes by substituting the event's BAP.
- **Immediate assessment priority remains Written Test 1.** It is on **Wednesday, October 28** during the enrolled lab session. Jackie directed students to the guide, example questions, and posted practice-question walkthrough for preparation.

Roadmap

1. Written Test 1, Lab 3, and review resources

Jackie began with the near-term priorities. Written Test 1 is scheduled for **Wednesday, October 28** during each student's enrolled lab session. Students were asked to keep Lab 3 on track. The Programming Test was still being graded.

For Written Test 1 preparation, Jackie pointed students to the guide, example questions, and the newly posted practice-question solution walkthrough on eClass. On October 22, Jackie confirmed a Zoom review session for Sunday, October 25; at the time of the lecture, its exact start time had not yet been confirmed. The intention was to use the prerecorded walkthrough for the practice-question solutions and reserve the live session for follow-up and conceptual questions.

2. Safe state versus enabled event

The first technical development revisited the M_0 Bridge Controller. Jackie separated two concerns:

- **Safety of a state:** evaluate the invariants only.
- **Enabledness of an event:** evaluate that event's guard in the current pre-state.

The example on p. 3 of Jackie's annotated iPad notes used

$$d = 3, \qquad n = 0, \qquad I : \ n \leq d,$$

and an illustrative guard for `ML_out`,

$$G : \ n \leq 1,$$

with action $n := n + 1$. The trace reaches $n = 0$, then $n = 1$, then $n = 2$. All three states satisfy the invariant, but at $n = 2$ the guard is false, so `ML_out` is disabled. The hypothetical state $n = 3$ would still satisfy $n \leq d$, yet the chosen guard makes it unreachable. The design requirement is that reachable states remain safe; it is not necessary that every safe state be reachable.

Jackie briefly contrasted this theorem-proving direction with model checking, where a complete transition graph may be explored algorithmically. That comparison was background only; this course proceeds with deductive proof and sequent calculus.

3. Before-after predicates and prime notation

Page 4 of Jackie's annotated iPad notes introduces the notation needed to talk precisely about a state transition. For a variable x:

- x denotes its **pre-state** value;
- x' denotes its **post-state** value.

Constants do not change across a transition, so there is no corresponding primed version of a constant such as d.

The Event-B action

$$n := n + 1$$

is action syntax. For mathematical reasoning, it is reformulated as the before-after predicate

$$n' = n + 1.$$

Jackie used **BAP** as the course abbreviation for a before-after predicate. This conversion is essential because the action syntax itself is not the predicate manipulated during a paper proof.

4. Why sequential programming intuition gives the wrong swap

The swap example on p. 5 makes the semantics concrete. Consider the familiar imperative-looking actions

$$\begin{aligned} temp &:= x, \\ x &:= y, \\ y &:= temp. \end{aligned}$$

Their BAP is

$$temp' = x \wedge x' = y \wedge y' = temp.$$

All right-hand sides refer to the *pre-state*. With $x = 23$, $y = 46$, and $temp = -1$, the post-state becomes

$$temp' = 23, \qquad x' = 46, \qquad y' = -1,$$

so x and y are not swapped.

The correct Event-B specification does not need a temporary variable:

$$x := y, \qquad y := x,$$

with BAP

$$x' = y \wedge y' = x.$$

The two constraints are satisfied simultaneously. Jackie's explicit warning was not to read Event-B action blocks as sequential Java execution.

5. Invariant preservation as a universal claim

Page 6 returns to the invariants of M_0. The state-space discussion here uses valuations satisfying the typing constraints, such as $n \in \mathbb{N}$. The safety property $n \leq d$ is then something that must hold for the relevant valuations.

The proof perspective is universal: the desired invariant must hold across all states covered by the claim. Therefore, the familiar quantifier strategy still applies: to refute a universal invariant claim, one concrete witness valuation that satisfies the setup but violates the invariant is enough.

6. Sequent syntax and semantics

Page 7 introduces the central proof notation. A sequent may be written horizontally as

$$H \vdash G$$

or in a vertical layout. The material to the left (or above) the turnstile is the set of assumptions/hypotheses; the material to the right (or below) is the goal/conclusion. Semantically, the hypotheses are combined and are intended to entail the goal.

Jackie illustrated non-provability with

$$d \in \mathbb{N}, \quad n \in \mathbb{N}, \quad n \leq d \ \vdash \ n + 1 \leq d.$$

Taking $d = 10$ and $n = 10$ satisfies all three hypotheses but makes the goal $11 \leq 10$, which is false. The valuation is therefore a witness that the sequent cannot be proved from those hypotheses.

A second point concerned an empty hypothesis set. Writing only

$$\vdash G$$

means proving G without extra assumptions. The empty conjunction behaves as `true`; treating it as `false` would make $\text{false} \Rightarrow G$ true for every arbitrary goal and would trivialize the proof system.

7. Sketching the invariant-preservation PO/VC for `ML_out`

The final annotated page gives a *sketch*, not yet the full inference-rule system. Jackie used **PO** (proof obligation) and **VC** (verification condition) for the condition that must be shown for verification. For invariant preservation, each event is treated separately.

For `ML_out`, the hypotheses are assembled from:

- the axiom $d \in \mathbb{N}$;
- the pre-state invariants $n \in \mathbb{N}$ and $n \leq d$;
- the guard of `ML_out`, which in the original M_0 model is `true` because there is no guard.

The intermediate post-state goal is

$$n' \in \mathbb{N} \wedge n' \leq d.$$

This is not the final paper sequent because the prime must be eliminated. Using the BAP

$$n' = n + 1,$$

substitute $n + 1$ for every occurrence of n'. The goal becomes

$$n + 1 \in \mathbb{N} \wedge n + 1 \leq d.$$

Thus the construction exposes exactly what must be justified for `ML_out` to preserve the invariants. Formal manipulation by inference rules was deferred to a later lecture.

Key Terms, Notation, and Structures

Safe state
A state whose invariant predicates evaluate to true. In the running example, $n \leq d$ is the safety invariant being checked.

Guard / enabling condition
A predicate evaluated in the current pre-state. The event is enabled when its guard is true. A guard restricts which transitions are allowed; it does not by itself determine whether a state is safe.

Pre-state and post-state
For a variable x, the lecture uses x for its value before a transition and x' for its value after the transition. Constants do not receive primes.

Before-after predicate (BAP)
A predicate relating pre-state and post-state values. Example: $n := n + 1$ is reformulated as $n' = n + 1$. "BAP" is the shorthand Jackie used in this course.

Parallel/simultaneous action semantics
Multiple Event-B actions constrain the post-state simultaneously. Their right-hand sides refer to pre-state values; they are not executed line by line as Java assignments.

Invariant preservation
The verification requirement that an event starting from a state satisfying the assumed model conditions leaves the relevant invariants true in the post-state.

Sequent and turnstile
A proof statement of the form $H \vdash G$, with hypotheses/assumptions on the left and the goal/conclusion on the right. The lecture related its meaning to implication from the hypotheses to the goal.

Empty hypothesis set
A sequent $\vdash G$ means G must be established without additional external assumptions. The absent hypotheses behave as `true`, not `false`.

PO / VC
Proof obligation / verification condition. In this lecture, the main construction was an event-specific invariant-preservation condition expressed as a sequent.

Key Examples and Demonstrations

1. Conservative guard: safe but unreachable

Setup: $d = 3$, $n = 0$, invariant $n \leq d$, guard $n \leq 1$, action $n := n + 1$.

Main point: The states $n = 0, 1, 2$ reached by the trace are safe, but at $n = 2$ the event is disabled. The state $n = 3$ is also safe with respect to the invariant, yet the chosen guard prevents reaching it.

What to practise: For each state in the diagram on p. 3 of Jackie's annotated iPad notes, evaluate the invariant and the guard separately. Do not use the guard to decide whether the state itself is safe.

2. The swap counterexample

Setup: $x = 23$, $y = 46$, $temp = -1$, with the three imperative-looking actions from p. 5.

Main point: Convert the actions to $temp' = x \wedge x' = y \wedge y' = temp$. The post-state is $(x', y', temp') = (46, -1, 23)$, not a swap. The corrected specification is $x' = y \wedge y' = x$.

What to practise: Translate each action into its BAP before evaluating post-state values. Verify that every right-hand side uses the pre-state.

3. Refuting a sequent with a witness valuation

Setup:

$$d \in \mathbb{N},\ n \in \mathbb{N},\ n \leq d \ \vdash\ n + 1 \leq d.$$

Main point: $d = 10$, $n = 10$ makes all hypotheses true and the goal false. The example reuses the proof/disproof strategy from the mathematical review: one witness is enough to refute a universal entailment claim.

4. Constructing the `ML_out` invariant-preservation condition

Setup: Take the M_0 axiom, the two pre-state invariants, the `ML_out` guard, and the event action.

Skeleton:

1. Collect the hypotheses: $d \in \mathbb{N}$, $n \in \mathbb{N}$, $n \leq d$, and `true`.
2. Write the post-state invariant goal: $n' \in \mathbb{N} \land n' \leq d$.
3. Convert the event action into the BAP $n' = n + 1$.
4. Substitute the BAP into the goal to obtain $n + 1 \in \mathbb{N} \land n + 1 \leq d$.

What to practise: Reproduce the color-coded pre-state/post-state construction on p. 8 of Jackie's annotated iPad notes without looking at the finished annotations, and explain why the final sequent should not contain primed variables.

Common Pitfalls and Misconceptions

- **Using a guard to decide whether a state is safe.** Safety is checked against the invariants; a guard only decides whether a particular event is enabled in that state.
- **Assuming every safe state must be reachable.** Jackie's $n \leq 1$ guard example deliberately blocks the safe state $n = 3$. A restrictive guard may be a design choice.
- **Reading $:=$ as sequential assignment.** In Event-B, the actions specify a transition relation. Multiple actions constrain the post-state simultaneously; Java-style line-by-line reasoning can give the wrong result.
- **Putting primes on constants.** Prime notation distinguishes pre-state and post-state values of variables. A constant such as d does not change across the transition.
- **Manipulating event action syntax directly in a proof.** First reformulate the action as a BAP such as $n' = n + 1$; the predicate is what is used for mathematical substitution and reasoning.

- **Treating empty sequent hypotheses as `false`.** That would make every goal trivially follow from a false antecedent. No hypotheses means the goal must be proved without extra assumptions.
- **Leaving primed variables in the final PO/VC sequent.** The post-state form with primes is an intermediate step. Use the event's BAP to eliminate the primes before the subsequent sequent-calculus proof.
- **Combining events into one invariant-preservation check.** Jackie emphasized that the invariant-preservation condition is formulated separately for each event, such as `ML_out` and `ML_in`.

What You Should Be Able to Do

After this session, you should be able to:

- distinguish a state's invariant-based safety from an event's guard-based enabledness;
- trace the guarded $d = 3$ example and identify a state that is safe but unreachable under the chosen guard;
- explain the pre-state/post-state convention x versus x' and why constants are not primed;
- translate an Event-B action such as $n := n + 1$ into the BAP $n' = n + 1$;
- explain why multiple Event-B actions are simultaneous constraints rather than sequential Java assignments;
- use the swap example to show why the three-line temporary-variable version fails as a formal specification and why $x' = y \wedge y' = x$ expresses the intended swap;
- read a sequent $H \vdash G$ by identifying its hypotheses and goal, and relate it to implication;
- explain why an empty hypothesis set behaves as `true`;
- refute an invalid sequent by finding a valuation that makes all hypotheses true and the goal false;
- construct the invariant-preservation PO/VC skeleton for one event by collecting axioms, pre-state invariants, and guards, writing post-state invariants with primes, and eliminating those primes through BAP substitution;
- state the confirmed Written Test 1 logistics: October 28 in the enrolled lab session, and identify the guide, example questions, and posted walkthrough as preparation resources.

Suggested Practice

1. **Written Test 1 preparation:** use the guide, example questions, and the posted practice-question solution walkthrough. Revisit the mathematical review material, especially the practice item on proving and disproving universal and existential claims with witness values.

2. **Redo the safe/enabled trace:** from p. 3 of Jackie's annotated iPad notes, evaluate $n \leq d$ and $n \leq 1$ separately at each displayed state. Explain why $n = 3$ is safe but unreachable in that design.

3. **Practise BAP translation:** translate the displayed actions for `ML_out`, `ML_in`, and the swap example into before-after predicates. Check carefully which symbols are pre-state values and which are post-state values.

4. **Reproduce the swap demonstration:** using $x = 23$, $y = 46$, $temp = -1$, compute the post-state from the three-line BAP and then compare it with the correct two-variable swap.

5. **Read and refute a sequent:** return to the $d = 10$, $n = 10$ witness for the example $n+1 \leq d$. Explain precisely which hypotheses remain true and why the goal fails.

6. **Reconstruct the `ML_out` PO/VC sketch:** begin with the model components, write the primed post-state invariants, then use $n' = n + 1$ to obtain the unprimed goal. Be comfortable with this substitution before the next lecture's inference-rule work.

Where to Review

- **Jackie's annotated iPad notes, p. 2:** Written Test 1, Lab 3, and the October 22 announcement of a Zoom review on Sunday, October 25. The start time was still to be confirmed when those notes were made.

- **Jackie's annotated iPad notes, p. 3:** "Safe State vs. Enabled Event" and the $d = 3$ guarded transition example, including the safe-but- unreachable state.

- **Jackie's annotated iPad notes, p. 4:** before-after predicates, pre-state/post-state notation, and the conversion from $n := n + 1$ to $n' = n + 1$.

- **Jackie's annotated iPad notes, p. 5:** the swap example contrasting imperative-looking action syntax with simultaneous BAP semantics.

- **Jackie's annotated iPad notes, p. 6:** invariant preservation as a universal condition over typed valuations and the witness strategy for refutation.

- **Jackie's annotated iPad notes, p. 7:** sequent syntax, the turnstile, hypotheses versus goals, the invalid $n + 1 \leq d$ example, and empty hypotheses.

- **Jackie's annotated iPad notes, p. 8:** the invariant-preservation PO/VC sketch for `ML_out`, including the pre-state/post-state versions of the invariants and BAP substitution.
- **eClass Written Test 1 materials:** the guide, example/practice questions, and the posted practice-question solution walkthrough referenced by Jackie at the beginning of the lecture.

Closing Reminder

Use this digest as a roadmap for active review, not as a substitute for the learning process itself. Attend Jackie's in-person sessions and pay attention to how he develops ideas and works through the important reasoning step by step. If a transition, explanation, example, or reasoning step remains unclear afterwards, return to the recording to fill that gap.

For structured review, start with Jackie's annotated iPad notes because they most closely reflect the actual flow and development of the class. Work through them in manageable logical units. After each logical unit, study the corresponding part of the presentation slides alongside it. Jackie does not normally teach by reading the slides line by line, so the slides may contain relevant supporting material that was not explicitly read aloud.

Finally, choose a suitable selection of the important reasoning from this session and work through it again on your own. The goal is not to memorize Jackie's recorded steps, but to determine whether you can reproduce the reasoning from your own understanding and explain why the important steps work.