EECS3342 System Specification and Refinement
Fall 2026

Digest of Lecture 10, October 20

Bridge Controller: Correct by Construction and M0

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

- **The course moved from mathematical review to the Bridge Controller case study.** Lecture 10 established the background for using formal models, requirements, invariants, proof obligations, and refinement over the remainder of the term rather than trying to encode the whole system at once.
- **Correct by construction means building a sequence of models.** Start with a simple, highly abstract model and add detail through refinements. Each refinement step will eventually have to be justified by proof obligations; a single all-encompassing "Superman model" is intentionally avoided.
- **A model's state space consists of valuations that satisfy its constraints.** Constants, variables, typing constraints, and safety invariants jointly determine which configurations are legitimate. Explicitly enumerating all configurations is generally infeasible, motivating proof-based reasoning.
- **M0 uses a deliberately coarse abstraction.** For the initial Bridge Controller model, the island and bridge are treated as one island-bridge compound. The model tracks only the aggregate number of cars in that compound, subject to a capacity bound.
- **Dynamic safety must hold across traces, not only initially.** The initial M0 events `ML_out` and `ML_in` are guardless and therefore always enabled. Concrete traces show that they can violate the invariants; one witness trace is enough to disprove the universal safety claim.
- **Immediate assessment priority: Written Test 1.** It is scheduled for **Wednesday, October 28** and covers the mathematical review material from before reading week, not the new Bridge Controller material introduced in this lecture.

Roadmap

1. Immediate course priorities after reading week

Jackie opened by marking the transition into the second half of the course. Lab 3 had been released. For the October 21 lab session, students could still use the lab room, but no TA would be physically present; remote TA assistance was available through the contact information on eClass.

The next major assessment is **Written Test 1 on Wednesday, October 28**. Jackie emphasized that it covers only the mathematical review completed before reading week. The guide and practice questions were already available, and the recommended preparation sequence was to revisit the relevant labs, slides, and underlying concepts. Jackie described the test as mostly multiple choice, with some written work possible. An online Zoom review session was planned for Sunday at roughly 5:30 p.m., with a previously recorded practice-question review to be posted beforehand.

2. What the Bridge Controller case study is meant to teach

The new case study will be used to connect the earlier mathematics to formal specification and refinement. Jackie previewed several ideas that will be introduced gradually: requirements expressed through environmental and functional descriptions, contexts and machines, invariants and safety properties, deadlock freedom, liveness, and proof obligations.

The proof work will be developed using *sequent calculus* and inference rules rather than relying only on Rodin. Lecture 10 only previewed this proof framework; the formal proof machinery comes later.

3. Correct by construction and the refinement pipeline

The central diagram on p. 3 of Jackie's annotated iPad notes represents a sequence

$$M_0, \ldots, M_i, M_j, \ldots, M_n.$$

The purpose is to avoid constructing and verifying one "Superman model" that contains every variable, detail, and requirement at once. Instead, properties are distributed across different models and detail is added progressively.

The terminology developed in class was:

- M_i is *refined by* M_j, or equivalently M_j *refines* M_i;
- the earlier model is more abstract and contains less detail;
- the later model is more concrete and contains more detail; and
- refinement is transitive, although formal verification normally focuses on consecutive refinement steps.

Jackie compared the idea loosely with increasing detail in software design, but explicitly distinguished formal refinement from object-oriented inheritance. The Bridge Controller used in this course will focus on M_0, M_1, and M_2.

4. State space: variables, constants, constraints, and valuations

Page 4 of Jackie's annotated iPad notes defines the state space as the set of all possible valuations of the declared constants and variables that satisfy the declared constraints. Jackie separated two important kinds of constraints:

- **typing constraints**, such as stating that a value belongs to a particular mathematical set; and
- **safety/invariant constraints**, which restrict acceptable system states beyond typing alone.

The course-level intuition is that, as refinements introduce more detail, the model becomes more concrete and its state space typically becomes larger and more complicated.

5. Bank-account valuation example

To make the state-space idea concrete, Jackie reused a Lab 1-style example with constants C and L and a variable $accounts$, where the constants give balance bounds and $accounts$ is a partial function. A desired property requires every defined account balance to lie between $-C$ and L.

One legitimate valuation used positive values such as $C = 10000$ and $L = 20000$ with $accounts = \varnothing$. The empty partial function satisfies the universally quantified balance property because there is no violating account. A non-empty valuation, such as an account for "Bill" with balance 5000, can also be legitimate when the bounds are satisfied.

The point was not to enumerate all possibilities. The set of admissible valuations is enormous, so formal proofs and proof obligations are used to reason universally about the state space.

6. Distributing Bridge Controller requirements across models

The requirements page on p. 5 distinguishes environmental (E-) descriptions from functional (R-) descriptions/properties and shows why they are not all placed into the same model. Jackie organized the intended development as follows:

- M_0 begins with the requirement that the number of cars on the bridge and island is limited;
- M_1 later adds the physical bridge direction constraints; and
- M_2 later introduces environmental assumptions involving the two traffic lights and driver compliance.

The sensor/actuator requirements shown in the requirements document were identified as beyond the main scope of this course. Jackie also stressed that an environment assumption such as cars obeying a red light can be essential to proving later safety properties.

7. M0 abstraction: the island-bridge compound

The first model deliberately ignores the internal distinction between island and bridge. As shown on p. 6 of Jackie's annotated iPad notes, the two are collapsed into a single *island-bridge compound*. Jackie used an "airplane view" intuition: from a sufficiently abstract viewpoint, the bridge is not distinguished from the island.

The variable n represents the *aggregate* number of cars in this compound. If reality has 100 cars on the island and 3 on the bridge, then the M0 value is simply

$$n = 100 + 3 = 103.$$

M0 does not record how those 103 cars are split between island and bridge. That extra detail belongs to a later refinement.

8. Static and dynamic parts of M0

Page 7 separates the static context from the dynamic machine. The context contains the capacity constant d with axiom

$$\texttt{axm0_1}: \quad d \in \mathbb{N}.$$

The machine contains the current aggregate count n together with invariants

$$\texttt{inv0_1}: \quad n \in \mathbb{N}, \qquad \texttt{inv0_2}: \quad n \leq d.$$

The mainland count is deliberately not modeled or bounded in M0. Jackie also introduced the naming convention in which the model index and invariant/axiom number are carried in labels such as `axm0_1` and `inv0_2`.

9. M0 events, guardless enablement, and unsafe traces

The dynamic behavior on p. 8 uses two events:

$$\texttt{ML_out}: \quad n := n + 1, \qquad \texttt{ML_in}: \quad n := n - 1.$$

`ML_out` represents a car leaving the mainland and entering the island-bridge compound; `ML_in` represents a car entering the mainland from the compound. Neither event has a guard, so both are always enabled.

This immediately creates safety problems. With $d = 2$ and initial $n = 0$, executing `ML_in` produces $n = -1$, violating $n \in \mathbb{N}$. A longer witness trace

$$0 \xrightarrow{\texttt{ML_out}} 1 \xrightarrow{\texttt{ML_out}} 2 \xrightarrow{\texttt{ML_out}} 3$$

violates $n \leq d$ at the final state.

Jackie then connected this to the earlier logic review: the intended safety claim is universal—every valid trace, meaning every sequence of enabled events, should end in states satisfying the invariants. To disprove that claim, one witness trace leading to an invariant violation is enough.

Key Terms, Notation, and Structures

Correct by construction
A development strategy in which a system is built as a sequence of increasingly concrete models and each refinement step is justified by proof obligations.

Refinement
Moving from an earlier, more abstract model to a later, more concrete model by adding detail. In the lecture's terminology, M_i is refined by M_j, or M_j refines M_i.

Abstract versus concrete
Abstract means less detail; concrete means more detail. The course starts with a simple abstract model and progressively approaches a model sufficiently close to implementation.

State space
The set of possible valuations of a model's declared constants and variables that satisfy the declared constraints.

Typing constraint versus safety invariant
A typing constraint states what mathematical set a value belongs to. A safety invariant restricts which typed states are acceptable for the intended system.

Context and machine
The M0 context contains static information such as constant d and its axiom. The machine contains dynamic state such as variable n, its invariants, and the events that update it.

Island-bridge compound
M0's abstraction that treats the island and bridge as one unit. The aggregate car count n does not distinguish where cars are located inside that compound.

`axm0_1`, `inv0_1`, `inv0_2`
The naming pattern introduced for the first model's axiom and invariants. In M0, $d \in \mathbb{N}$, $n \in \mathbb{N}$, and $n \leq d$.

Guardless event
An event with no guard. Jackie emphasized that such an event is always enabled, which is dangerous when its action can move the model outside the invariant.

Valid trace
A sequence of enabled events. The safety idea developed in class requires every valid trace to preserve the invariants; a single violating trace is a witness against that universal claim.

Proof obligation
A formal statement that must be discharged to justify model correctness or a refinement step. Lecture 10 motivated proof obligations; later lectures develop their exact forms and the sequent-calculus method used to prove them.

Key Examples and Demonstrations

1. **The refinement pipeline (p. 3).**

   Setup. Compare one "Superman model" with a sequence $M_0, \ldots, M_n$.

   Main point. Begin with the simplest useful abstraction, distribute properties across models, and add detail step by step. Each arrow in the pipeline eventually corresponds to proof obligations that justify the refinement.

   What to practise. Given two consecutive models, state correctly which one refines the other and which is more abstract/concrete.

2. **Bank-system state-space valuations (p. 4).**

   Setup. Use positive constants C and L, a partial function $accounts$, and the invariant that every defined balance lies between $-C$ and L.

Main point. Both empty and non-empty account functions can be valid valuations when all constraints hold. The state space is too large to enumerate explicitly.

What to practise. Decide whether a proposed valuation satisfies the typing constraints and the safety property, and explain why $accounts = \varnothing$ satisfies the universal balance condition.

3. **The M0 aggregate-count abstraction (p. 6).**

   Setup. Reality has 100 cars on the island and 3 on the bridge.

   Main point. M0 records only $n = 103$. It intentionally does not record the internal split. That omission is the abstraction, not a missing feature to be repaired immediately.

   What to practise. Explain what information M0 preserves and what information it deliberately postpones to later refinements.

4. **Static M0 state space (p. 7).**

   Setup. Context constant d, machine variable n, with $d \in \mathbb{N}$, $n \in \mathbb{N}$, and $n \leq d$.

   Main point. The context provides static assumptions, while the machine contains changeable state and invariants. The mainland count is outside M0's modeled state.

   What to practise. Identify which statements are axioms and which are invariants, and explain the roles of d and n.

5. **Underflow witness from a guardless event (p. 8).**

   Setup. Start from $d = 2$, $n = 0$, with `ML_in` always enabled.

   Main point. One occurrence gives $n = -1$, immediately violating $n \in \mathbb{N}$.

   What to practise. State which invariant fails and why the lack of a guard permits the bad transition.

6. **Overflow witness trace (p. 8).**

   Setup. Start again from $d = 2$, $n = 0$ and repeatedly execute `ML_out`.

   Main point. The states 0, 1, and 2 are safe with respect to $n \leq d$, but the next transition reaches $n = 3$, which violates the capacity invariant.

   What to practise. Reproduce the trace and explain why one such witness is enough to disprove the statement that every valid trace preserves the invariants.

Common Pitfalls and Misconceptions

- **Do not build a "Superman model".** Jackie explicitly warned against putting every requirement and detail into one model and then trying to verify it directly. The case study is organized around staged refinement.
- **Do not equate refinement with inheritance.** The abstraction analogy can be useful, but formal refinement is about adding specification detail and discharging proof obligations, not object-oriented subclassing.

- **Do not branch the refinement pipeline in this course model.** Jackie responded to a student question by keeping the development sequential rather than creating parallel variants such as $M_{1.1}$ and $M_{1.2}$.
- **Do not interpret n as only the number of cars on the bridge.** In M0, n is the aggregate number of cars in the entire island-bridge compound.
- **Do not add mainland capacity constraints that M0 does not model.** The mainland count is unspecified/unbounded at this abstraction level.
- **Do not assume that a safe initial state makes the model safe.** Guardless events can move a safe state into an invariant-violating state. Safety has to survive every valid trace.
- **Do not confuse "enabled" with "must happen next".** With both M0 events enabled, different event sequences are possible. The correctness question is whether *any* valid sequence can reach a violation.
- **Do not carry later refinements into M0 prematurely.** Traffic lights, directional distinctions, sensors, and similar details are intentionally postponed; adding them now defeats the abstraction strategy.

What You Should Be Able to Do

After this session, you should be able to:

- explain why correct-by-construction development uses a sequence of models rather than one all-encompassing model;
- use the phrases "M_i is refined by M_j" and "M_j refines M_i" correctly and distinguish more abstract from more concrete models;
- explain conceptually why properties are distributed across refinements and why proof obligations are needed between refinement steps;
- describe a model's state space in terms of constants, variables, valuations, typing constraints, and invariants;
- distinguish a typing constraint from a safety invariant using the bank-account example;
- explain the M0 abstraction and compute the aggregate value n from a concrete island-plus-bridge count without introducing an internal split;
- identify the roles of d, n, `axm0_1`, `inv0_1`, and `inv0_2`;
- interpret `ML_out` and `ML_in` and explain why events without guards are always enabled;
- trace the M0 events from a concrete state and test the resulting state against the invariants;

- construct a witness trace that disproves the claim that all valid traces preserve the invariants; and
- state the scope of Written Test 1: mathematical review material from before reading week, not the Bridge Controller material introduced here.

Suggested Practice

1. **Prepare for Written Test 1.** Before October 28, work through the released guide and practice questions and revisit the relevant labs, slides, and mathematical-review concepts, as Jackie recommended. Use the Sunday Zoom review session for remaining questions after first attempting the materials.
2. **Recreate the refinement pipeline on p. 3.** Without looking, draw M_0 through M_n, label which direction increases concreteness, and state the two equivalent refinement phrasings for consecutive models.
3. **Rework the state-space example on p. 4.** For the displayed bank model, test the empty-account valuation and the non-empty example against both typing and safety constraints. Explain why enumeration is not a practical verification strategy.
4. **Rebuild the requirement distribution on p. 5.** Identify which requirement starts in M0 and which kinds of requirements are deliberately postponed to M1 and M2. Focus on the reason for distributing properties rather than memorizing the page layout.
5. **Redraw the M0 abstraction on p. 6.** Start from an island, bridge, and mainland picture, then deliberately collapse island and bridge into one compound and define what n represents.
6. **Reconstruct the static M0 model on p. 7.** Write the constant, variable, axiom, and two invariants from memory, including the label pattern.
7. **Reproduce both unsafe traces on p. 8.** Starting from $d = 2$ and $n = 0$, show the one-step underflow using `ML_in` and the repeated `ML_out` overflow. For each, state exactly which invariant becomes false.
8. **Review before the next lecture.** Jackie explicitly asked students to revisit the M0 abstraction diagram and distinguish its static and dynamic parts before Thursday's continuation.

Where to Review

- **Jackie's annotated iPad notes, p. 2:** Lab 3/Written Test 1 priorities and the October 28 test date.
- **p. 3:** correct-by-construction refinement pipeline, the "Superman model" warning, abstract/concrete terminology, and refinement wording.

- **p. 4:** state-space definition, the bank-account valuation example, typing versus safety constraints, and the reason to prove rather than enumerate configurations.
- **p. 5:** Bridge Controller requirements document and the staged distribution of requirements across M0, M1, and M2.
- **p. 6:** the M0 abstraction into an island-bridge compound and the aggregate-count example $100 + 3 = 103$.
- **p. 7:** M0 context/machine split, constant d, variable n, axiom/invariant labels, and the unmodeled mainland count.
- **p. 8:** `ML_out/ML_in`, lack of guards, state-transition traces, safe versus unsafe states, and the witness-trace idea.
- **Session recording:** return to the opening logistics for Written Test 1 preparation and review-session details; return to the later discussion for Jackie's explanations of refinement versus inheritance, the airplane-view abstraction, and the student question about always-enabled events.

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.