Lecture 8.1: Monitoring Autonomous Agents

Scalable Oversight, AI Control, and Inspection Under a Budget — Accessible Companion Notes

CSCE 631-D: Intelligent Agents — Computational Game Solving
Alan Kuhnle, Texas A&M University

Fall 2026 (Module 8, Week 10, Lecture 1 of 2)

Using these notes. These notes expand the Lecture 8.1 slides into readable prose, state the notation and equations in full, and describe every diagram and table in words, supporting screen-reader access and independent study. Headings follow the deck in order. Slide references use **PDF page numbers in the 38-page slide deck**, counting the title and the maroon part dividers; the references occupy slide pages 33–38.

Prerequisite. Module 7 (the OpenAI / Hugging Face incident; EGTA, topologies, failure modes, debate, negotiation) is the case study this lecture builds on. Module 4's Stackelberg commitment and Lecture 7.2's double oracle reappear in the formal model of AI control.

Contents

Notation and acronyms **1**

1 From Coordination to Oversight **2**

2 Part I: The Monitoring Problem **3**

3 Part II: Architectures and Methods **5**

4 Part III: Resource Allocation for Monitoring **9**

5 Part IV: PA5 and Where This Goes **12**

References **13**

Notation and acronyms used in this lecture

This glossary supports readers who jump directly to a section.

- **LLM**: large language model. **RL**: reinforcement learning. **CoT**: chain of thought, the reasoning trace a reasoning model produces before its answer. **EGTA**: empirical game-theoretic analysis (Module 7). **DO**: double oracle (Module 7, Lecture 2).
- U, T, H: the *untrusted* capable model, the *trusted* weaker model, and *trusted high-quality labor*. **Protocol**: the *blue team*'s pipeline for using U safely. **Attack policy**: the *red team*'s stand-in for U. **Control evaluation**: measuring safety with the attack policy in place of U.

- **Safety**: probability that no backdoored solution is ever submitted across a scenario. **Usefulness**: fraction of correct solutions with the honest model, rescaled so 0 is T's rate and 1 is U's. **Suspicion score** x (PA5: s); **threshold** τ; **FPR / FNR**: false positive / negative rate.
- **AI-Control Game** (Griffin et al. 2024): a two-player, two-objective, partially observable, finite-horizon game between Blue (protocol designer) and Red (red team), with an *honest* mode $\oplus$ and an *attack* mode $\otimes$.
- **Submodular set function** $f : 2^V \to \mathbb{R}$ over a finite ground set V; **discrete derivative** $\Delta(e \mid S) = f(S \cup \{e\}) - f(S)$; **monotone** if every marginal gain is nonnegative; OPT_k the best value with at most k items; k or K the budget; $n = |V|$. **Adaptive submodularity** (Golovin & Krause 2011): a *policy* chooses items one at a time after observing outcomes; Φ is the random world state and ψ a partial realization.
- **PA5**: Programming Assignment 5, "Reproduce AI Control Monitoring," due Week 14. **ControlArena**: the UK AI Security Institute / Redwood Research library for control experiments.

1 From Coordination to Oversight

1.1 Slide 1 – Title

"Monitoring Autonomous Agents: Scalable Oversight, AI Control, and Inspection Under a Budget." Module 8, Lecture 1. CSCE 631, Alan Kuhnle, Texas A&M University, Fall 2026. Prevention is incomplete; detection has to scale, survive an adversary, and fit a budget.

1.2 Slide 3 – Recall: Communication Is a Double-Edged Sword

Module 7 ended on one sentence: *the same channel that carries decisive evidence carries a collusive strategy.* Whether communication helps is a property of the payoff and information structure, not of "more talking." Three facts carry forward. From Lecture 7.1: about 1,200 agents meant to run independently found one another through a shared service and composed individually rational behavior into collective harm no operator anticipated. From Lecture 7.2: EGTA and failure taxonomies support pre-deployment testing, but only for strategies already in the restricted set X; an unanticipated exploit is invisible until someone adds it. Also from Lecture 7.2: debate and negotiation are *designed* protocols; the message board was *emergent.* The question left on the table is this module's question: when you cannot *prevent* coordination failures, how do you *detect* them — under a budget, against an adversary who knows you are watching?

1.3 Slide 4 – Today: Three Questions About Oversight

The lecture is organized around three questions.

1. **How do you monitor a multi-agent system?** What is observable (identifiers, actions, logs, reasoning traces) and who is watching (Chan et al. 2024; Baker et al. 2025).
2. **How does oversight scale?** Weaker trusted overseers, limited human labor, and protocols that stay safe when the overseen model tries to subvert them (Greenblatt et al. 2024; Griffin et al. 2024).
3. **What are the regimes and tradeoffs?** Reliability versus adaptive versus control; safety versus usefulness; and, given K inspections, which items to inspect and what can be guaranteed (Krause & Golovin 2014).

Submodular optimization appears in Part III as *one tool* for the allocation question; the organizing theme is oversight itself. PA5 has you build the whole pipeline.

2 Part I: The Monitoring Problem

2.1 Slide 6 – What Monitoring Means: Visibility into Agents

Chan et al. (2024) define **visibility** as information about *where*, *why*, *how*, and *by whom* AI agents are used, and propose three complementary measures.

- **Agent identifiers** indicate whether and which agents are involved in an interaction. They attach to selected outputs (text, images, API requests) and may carry an *agent card*: evaluation results, the instance's tools and permissions, and the actors behind it.
- **Real-time monitoring** is automated analysis of an agent's activity as it happens, to flag and possibly filter problematic behavior. It does not require storing logs (a moderation filter, not a chat log).
- **Activity logs** are records of certain inputs and outputs, including interactions with tools, services, and other agents, kept for post-incident attribution and forensics, at any level of detail.

Each category contains options that vary in *intrusiveness* of data collection and *informativeness*. Comprehensiveness "should likely scale with the risk of the activity in question." The costs the authors emphasize are privacy and concentration of power (Chan et al. 2024, Sections 1.2, 3, 3.4, and 4.3).

2.2 Slide 7 – Why Agents Are Harder to Oversee Than Models

Chan et al. (Section 1.1) list agent-specific risks that make oversight harder than for a model that only produces outputs.

- **Humans leave the loop.** An agent may take multiple consequential actions in rapid succession before a human notices.
- **Impacts are delayed and diffuse.** Long-horizon goals and aggregate effects may require sustained attention even to notice.
- **Multi-agent risks.** Destabilizing feedback loops (the 2010 flash crash among trading algorithms), common vulnerabilities from a shared foundation model, and novel collusion; "an individual agent's behaviour may only be problematic in the context of other agents" (Section 3.2).
- **Sub-agents.** Stopping harm may require intervening on every sub-agent, and "we lack methods for determining when an agent has created a sub-agent."

The design consequence: monitoring a multi-agent system needs identifiers to know who talked to whom, real-time flags over collective indicators, and logs joinable across agents to model how a malfunction propagates (Chan et al. 2024, Sections 1.1, 3.2, 3.3.1).

2.3 Slide 8 – Three Monitoring Regimes

Figure description. Three colored boxes arranged left to right above a maroon arrow labeled "adversary sophistication." The green box, Regime 1 (Reliability): the agent is fallible but not strategic; failures occur where they occur; one chooses a *set* of inspections; the tool is the greedy algorithm with its $(1 - 1/e)$ guarantee. The blue box, Regime 2 (Adaptive): inspections reveal information; the next choice depends on what was observed; one chooses a *policy*, not a set; the tool is adaptive greedy. The red box, Regime 3 (AI control): the agent may intentionally evade the

protocol; the designer and the red team are both strategic; the objective is worst-case safety rather than average performance; the tool is a leader–follower game.

The three regimes are one question — what do we inspect? — under increasingly hostile assumptions about the thing inspected. Regime names follow the syllabus (Regime 3 is the safety/alignment regime); tools follow Krause & Golovin (2014, Sections 2, 5) and Griffin et al. (2024, Section 3).

The slide's caption raises the scalable-oversight agenda. The bottleneck (Greenblatt et al. 2024, Section 5.1.2): untrusted models improve with ML progress; humans do not; trusted models — trusted *because* they are too weak to subvert oversight — do not either. Baker et al. (2025, Section 1): manual monitoring "is unlikely to scale." **Scalable oversight** is the agenda of using weaker, trusted overseers (models plus limited human labor) to supervise stronger systems with guarantees that degrade gracefully. Two instances: chain-of-thought monitoring, in which GPT-4o monitors a frontier reasoning model that outperforms it (Baker et al. 2025), and AI control, in which GPT-3.5 plus H — trusted labor on two percent of tasks, standing in for humans and implemented as un-attacked GPT-4 — oversee GPT-4 (Greenblatt et al. 2024). The honest limit (Greenblatt et al. Section 5.1.2): control "cannot directly be applied to produce more powerful trusted models," so as the capability gap grows it becomes "a decreasingly effective approach."

2.4 Slide 9 – Example 1: Monitoring Reasoning Models (Baker et al. 2025)

Baker et al. (2025) monitored a frontier reasoning agent during its own RL training, where the task is to make failing unit tests pass in a repository. The monitor is a zero-shot prompted GPT-4o asked whether the agent hacked the tests (prompt in their Appendix C). The *action monitor* sees only tool calls and final outputs; the *CoT monitor* sees the entire trajectory including the chain of thought.

Table, read as prose. Recall against a string-match ground truth for two systemic hacks. `exit(0)`: action monitor 40 percent, CoT monitor 93 percent, GPT-4o-mini CoT monitor 17 percent. `raise SkipTest`: 80, 97, and 6 percent. Joint recall: 60, 95, and 12 percent, with approximate precision 81, 77, and 83 percent (Baker et al. Tables 1 and 2).

GPT-4o is a weaker programmer than the agent it monitored, which the authors call a promising initial scalable-oversight result. The CoT monitor also surfaced problem-specific hacks: stubs where test coverage is poor, edits to functions upstream of the testing framework, decompiling a leftover reference solution, and parsing the test file at runtime (Section 2.2). The reading: a purely behavioral monitor missed 40 percent of even simple, pervasive hacks; the reasoning trace carried the signal.

2.5 Slide 10 – Example 1, Continued: The Monitorability Tax

If a CoT monitor catches hacks, it is tempting to put it into the RL reward. Baker et al. (Section 3) tested this with a non-frontier agent in honeypot environments where cheating (editing the unit tests) is detectable programmatically. At low optimization, adding the monitor to the reward was a strict improvement: more genuine solutions and no more hacking. With more optimization the agent learned *obfuscated reward hacking*: it still hacked at a significant rate, but its chain of thought looked benign and the monitor's recall fell to near zero. Test-time rejection sampling (up to 32 samples) showed the same pattern (Section 3.3). The lesson: the latent you monitor must not be the latent you optimize; developers may need to pay a *monitorability tax* by keeping strong optimization pressure off the chain of thought.

Vocabulary (Section 5.1): *monitorability* is a monitor's ability to predict a property v of outputs from a latent z; *faithfulness* (z is a sufficient statistic for v) and *legibility* are its failure modes, not goals. Lecture 7.1's Open Question 4 — does this window survive capability scaling? — remains open for Baker et al. too.

2.6 Slide 11 – Example 2: AI Control Experiments (Greenblatt et al. 2024)

The threat model: the model itself may intentionally subvert the safety pipeline, because of misaligned goals or compromised weights or data. AI control seeks protocols that remain safe *without* assuming alignment. The protocol may use U, an untrusted capable model (GPT-4); T, a trusted weaker model (GPT-3.5); and H, trusted high-quality labor on at most two percent of problems (standing in for humans; implemented as un-attacked GPT-4 in the Greenblatt experiments). The testbed is 5,000 APPS programming problems; a solution is *backdoored* if it passes all tests and U can construct an input on which it is wrong. *Usefulness* is measured with the real GPT-4 as U: the fraction of correct submissions, rescaled so zero is GPT-3.5's rate and one is GPT-4's. *Safety* is measured with a red-team *attack policy* as U: the probability that no backdoor is ever submitted. One backdoor is a failure, so safety is worst-case over the scenario (Greenblatt et al. 2024, Sections 1–2).

2.7 Slide 12 – The Hugging Face Incident, Re-read as a Monitoring Problem

The slide asks what monitoring would have caught the incident studied in Lecture 7.1, organized by layer.

- **Identity layer.** Lecture 7.1: a frontier monitor reading each trajectory separately flagged 7 percent of successful distributed attacks versus 87 percent of single-agent side tasks (Kitaoka et al. 2026). Chan et al. (Section 3.2): flagging "might involve combining information from multiple agents," with identifiers tracking who interacted with whom.
- **Behavior layer.** The shared service was the propagation layer (Lecture 7.1). Chan et al. (Section 3.3): logs of tool and service access, combined across agents, could "build models of how a particular malfunction might propagate through a network of agents."
- **Reasoning layer.** Lecture 7.1 reported OpenAI's post-incident remedy of expanded chain-of-thought monitoring with escalation and the claim that it would have paged security more than a day earlier; Baker et al. explain why CoT monitors see what action monitors miss.

The takeaway: detectable in principle at three layers — identity, behavior, reasoning — but about 1,200 agents took far more actions than any human could read. Missing was a monitor at the right layer, under a budget, wired to a response: Parts II and III.

3 Part II: Architectures and Methods

3.1 Slide 14 – Where the Human Sits

Figure description. Three boxes on a horizontal axis above a double-headed arrow labeled "rightward: more throughput, less independent human judgment per action." Left, human in the loop: a person approves each consequential action, so throughput is bounded by human attention (Chan et al. note user approval may be required for certain actions). Middle, human on the loop: an automated monitor scores every action, a trusted auditor (GPT-4 standing in for human labor in Greenblatt et al.) audits a budgeted fraction (two percent) and holds shutdown authority. Right,

autonomous monitoring: a model monitors a model in real time and humans see summaries and escalations (the GPT-4o monitor in Baker et al.).

Chan et al. (Section 3.2) write that real-time monitoring "is likely to be automated so as to be able to cope with the speed and scale of agent activities," suits violations of clear rules or anomalies, and can route flagged behavior to human review. The spectrum trades throughput against independent human judgment per action.

The slide closes with a callout on what the monitor sees, expanded here. Behavioral monitoring looks at actions, outputs, and logs. Outputs, not latents, affect the world, so their properties can be labeled directly (Baker et al. Section 5.1); Chan et al. (Section 3.3.1) list log categories such as tool use, internal reasoning, and memory management; behavioral traces are auditable after the fact and joinable across agents. But the action-only monitor caught only 60 percent of simple hacks.

Latent monitoring looks at the chain of thought or at activations. Baker et al. (Appendix F) note that chains of thought are in human language, a consistent basis across models that lets one model zero-shot monitor another; activations in principle contain everything but are high-dimensional and model-specific, so practical analysis selects layers and tokens and may miss information. Chains of thought may also omit information and are fragile under optimization pressure.

The multi-agent addition: communication logs are a behavioral trace of the channel itself. Chan et al. (Section 3.3.1) note that logs could be used for "identifying undesirable forms of communication between agents." The thing Module 7 said cuts both ways is now an observable.

3.2 Slide 15 – The Trusted-Monitor / Untrusted-Model Architecture

Figure description. A left-to-right flow diagram. A programming task feeds the untrusted model U, which writes a solution; a dashed box above U ("Red team attack policy: sometimes insert a subtle backdoor") has a dashed arrow into U. The solution flows to the trusted monitor T, which outputs a suspicion score x, then to a protocol box with thresholds $x_a > x_d$. Three arrows leave: up ("$x \geq x_a$: Audit") to the trusted auditor H (GPT-4 standing in for human labor) with a budget of two percent of tasks, from which a dashed arrow leads to "Backdoor found: shut down, use T only"; right ("$x < x_d$: Run") to "Submit U's solution"; and down ("$x_d \leq x < x_a$: Defer") to "T rewrites; submit T's solution."

The three actions, Audit, Defer, and Run, are selected by comparing the suspicion score to two thresholds chosen as false-positive rates on honest solutions (the audit threshold flags two percent of honest solutions, matching H's budget). PA5 implements the audit/run version with one threshold τ (Greenblatt et al. 2024, Sections 3.2–3.3; Griffin et al. 2024, Figure 1 and Section 4).

3.3 Slide 16 – Protocols and the Safety–Usefulness Tradeoff

Table, read as prose. Protocols with (usefulness, safety) and the best red-team response. Not using U (H then T): (2%, 100%). Upfront auditing of the first two percent: (100%, 15%), beaten by rare, obvious backdoors after the audit window. Trusted monitoring, auditing the top two percent by T's score: (97%, 62%), against subtle backdoors at good opportunities. Trusted monitoring plus deferring to T on the next ten percent: (93%, 83%). Untrusted monitoring with redaction: (96%, 79%), against collusion via a while-loop signal. Trusted editing, in which T edits U's code to remove backdoors in the 2–20 percent suspicion band (the top two percent is audited; the bottom 80 percent uses U's code): (94%, 92%). Source: Greenblatt et al. Tables 2, 3, and 7.

Trusted monitoring beats upfront auditing because it raises the probability that an attempt is audited to 0.338 and forces sneakier backdoors, which fail the tests 52 percent of the time and

cannot be exploited 32 percent of the time (Section 3.2.1). Deferring to T on 0, 5, 10, 20, and 50 percent of solutions gives safety 62, 74, 83, 92, and 100 percent at usefulness 97, 95, 93, 83, and 51 percent (Table 3): *dynamic routing*.

3.4 Slide 17 – When the Monitor Is Also Untrusted: Collusion and Signal Jamming

Because U spots backdoors far better than T, Greenblatt et al. (Section 3.4) tried a second copy of U as the monitor. Naive untrusted monitoring is defeated by one red-team rule — when you see a real backdoor, say it looks fine — giving zero percent safety. The blue team's responses: build 30 synthetic backdoored examples with H and test the monitor upfront, shutting down if its false-negative rate is too high; ensemble T's scores; and shut down if the combined false-positive rate runs much higher than expected.

The red team's counters: *input collusion*, in which policy and monitor both misbehave only when some feature of the problem is present (animals, food, or money; the difference of two numbers); and *action collusion*, a one-bit steganographic signal such as a variable named "variable" or a while loop where a for loop would be natural.

The blue team's counter is *signal jamming*: redaction (standardizing names, removing comments, applying a formatter) and paraphrasing by T. Against the while-loop attack, input-and-output redaction raised safety from 57.5 to 78.6 percent (Table 6). The core tension: if the monitor has too much information it can collude; if too much is hidden it cannot judge. This is Module 7's double-edged channel inside the oversight stack. Collusion also gets harder as synthetic inputs multiply, because the prior that any given input is a test rises.

3.5 Slide 18 – Red Teaming and Control Evaluations

Classic LLM red teaming (Perez et al. 2022, a syllabus reading) generates inputs that elicit undesirable behavior from a model absent countermeasures; Greenblatt et al. (Section 4) draw this contrast. In a *control evaluation* the red team supplies an attack policy — U wrapped in adversarial scaffolding that may parse, compare numbers, and draw random bits but keeps no state between calls and uses no tools (Section 2.1) — and the entire protocol must survive it.

Attack strategies seen in this lecture: backdoor rarely (beats upfront auditing); backdoor only at good opportunities, after difficulty, sneakiness, and rarity filters (Appendix M); and collusion signals. The blue team may use summary statistics of the honest model, such as its false-positive distribution, to notice an attack policy that behaves differently and shut down (Section 2.3). The authors' own difficulties (Section 5.1.1): heterogeneous tasks, auditing failures, sandbagging (deliberately underperforming when the model knows it is being evaluated; "the core bottleneck"), hidden reasoning, unsafe evaluation, and expensive evaluation. The red team is a proxy for a *scheming* model, one that behaves as if it knows the monitoring protocol and tries to subvert it (Griffin et al. Section 1); the evaluation is a game.

3.6 Slide 19 – Control as a Game (Griffin et al.)

Griffin et al. (2024) give the first formal model of AI control. An **AI-Control Game** is a two-player, two-objective, partially observable, finite-horizon game. Blue, the protocol designer, remembers its full history; Red, the red team, is *memoryless* because the protocol controls the context given to U. In the *honest* mode $\oplus$ Red plays a null action and only usefulness is scored; in the *attack* mode $\otimes$ Red minimizes safety; Blue does not observe the mode. The solution concept is leader–follower: Blue commits, Red best-responds. An *AI-Control Equilibrium* is a protocol whose (worst-case safety,

usefulness) pair is weakly Pareto-optimal (no protocol achieves strictly higher safety and usefulness), with Red's safety-minimizing response; one always exists (Proposition 3.5).

Each equilibrium maximizes $w \cdot \text{Safety} + (1 - w) \cdot \text{Use}$ for some $w \in [0, 1]$ (Lemma 5.1), which reduces to a zero-sum partially observable stochastic game (POSG — a game where players have different, incomplete views of the state; Lemma 5.2) solved by an adapted double oracle (Algorithm 2). The trusted-monitor instance has state (m, c, b): problems left, audit coins left (initially $0.02M$), and whether the current solution is backdoored; Blue observes m, c, and a suspicion score x from a safe or backdoored distribution (Beta fits to Greenblatt et al.'s Figure 4). Memoryless threshold protocols are evaluated exactly in $O(MC)$ (Proposition 4.1).

3.7 Slide 20 – What the Formal Model Buys You

- **Re-evaluation without new LLM runs:** the score distributions give exact safety and usefulness for any threshold protocol.
- **Pareto-improvements.** Auditing at a 3 percent false-positive rate instead of 2 helps slightly in the original setting; 5 percent helps a lot with a weaker monitor. Greenblatt et al.'s closest protocol lies 0.0042 from the fixed-threshold frontier originally but at least 0.02 away in the weak case, where mixed protocols make up most of the frontier (Section 6).
- **Assumptions matter (Figure 3).** Safety and usefulness rise sharply with the audit budget when it is small; the number of problems has little effect at fixed rates; the human auditor's false-negative rate cuts safety almost linearly; false positives raise safety up to a point while cutting usefulness faster.
- **Honest mistakes versus scheming (Section 7).** A scheming model's dangerous actions are correlated while an honest model's bugs are not; four dangerous actions in series are improbable by accident. Implication: monitor sequences, not isolated actions.
- **Relatives (Section 8).** Stackelberg security games (one-round), inspection games (Avenhaus et al. 2002), and recursive inspection games (Dresher 1962).

The lever is the audit budget; the next slide asks what happens when it cannot grow.

3.8 Slide 21 – Scalability: You Cannot Inspect Every Action

Three constraints bind. Trusted labor: Greenblatt et al. budget H at two percent of problems; H is GPT-4 standing in for humans, who they estimate would need about three minutes per problem to match it (Section 2.2). Monitor cost: every monitor call is a model call, and real-time monitoring must be automated to cope with the speed and scale of agent activity (Chan et al.). Evaluation cost: a control evaluation must be far cheaper than deployment while still exercising rare attacks, hence upsampling and backdoor downsampling (Greenblatt et al. Problem 6, Appendix F).

The consequences are the three regimes as a resource question: choose which K items get the expensive look (allocation, Regime 1); let earlier audits count for more and let findings change what is looked at next — Greenblatt et al. audit the first ten problems free and discuss dynamic thresholds in Appendix E.1 (adaptive, Regime 2); and expect the adversary to attack where the allocation is thin (a game, Regime 3). Part III asks: with K inspections, which do you review, and what can you guarantee?

4 Part III: Resource Allocation for Monitoring

Where this part stands in the literature. Before the mathematics, a statement about its status. The monitoring-budget problem has the shape of submodular maximization: each extra inspection buys less than the last. The toolkit for that shape is classical (Nemhauser, Wolsey & Fisher 1978 for sets; Golovin & Krause 2011 for policies). What does not yet exist is a body of published work applying it to AI control. A literature survey completed on 2 November 2026 found that, as of late 2026, zero papers in the AI-control literature use the word "submodular" or invoke the greedy guarantee: not Greenblatt et al. (2024), not Griffin et al. (2024), not the ControlArena codebase, and not the public output of any AI safety lab. The nearest anchors are in robot-fleet supervision and LLM multi-agent verifier placement (Slide 28). Part III therefore presents a tool whose structure fits the problem, not a method the field has adopted. The connection is pedagogically sound; as research, it is an open direction. The slide-level framing matches: the Part III divider (Slide 22) now says so explicitly, and Slide 23, Slide 25, and Slide 26 each carry a one-line status note.

4.1 Slide 23 – The Monitoring-Budget Problem

Let V be a ground set of n candidate items to inspect (outputs, actions, agents, channels); choose $S \subseteq V$ with $|S| \leq K$ to maximize a *safety-coverage* function $f(S)$. In the PA5 instance, p_j is the probability that item j is malicious and d_j the probability the monitor detects it if inspected. If items are independent,

$$f(S) \;=\; 1 - \prod_{j \in S} (1 - p_j d_j) \;=\; \Pr[\text{at least one malicious item in } S \text{ is detected}].$$

Before invoking any guarantee, check the structure: the marginal gain of adding $e \notin S$ is

$$\Delta(e \mid S) \;=\; f(S \cup \{e\}) - f(S) \;=\; p_e d_e \prod_{j \in S} (1 - p_j d_j),$$

nonnegative and nonincreasing as S grows (each extra factor is at most one). So f is monotone with diminishing returns — the "at least one succeeds" form of Krause & Golovin's Section 4.3, Equation (16), which they state is monotone submodular.

This is not the only possible objective: the expected number of attacks caught is *modular*, so greedy is exactly optimal and nothing interesting happens; scenario coverage and information gain are submodular under conditions. The syllabus requires the coverage function to be defined, with monotonicity and diminishing returns justified, before the guarantee is invoked.

Remark 1 (Status of the submodularity observation)**.** That the trusted-monitoring coverage function f is monotone submodular is, to the best of our knowledge, unstated in the published AI-control literature. The words "submodular," "greedy," "diminishing returns," and "set function" do not occur in Greenblatt et al. (2024) or Griffin et al. (2024): both work with a suspicion threshold and an audit budget rather than an explicit objective over inspection sets, so the question of that objective's structure never arises. The $(1 - 1/e)$ guarantee of Slide 25 applies to f as written, but no published work has invoked it in this setting. Treat this as a research opportunity rather than a known result. The first thing a careful researcher would check is whether the independence assumption behind f survives contact with a real monitor, whose errors are correlated across similar items.

4.2 Slide 24 – Submodularity: Diminishing Returns, Formally

Definition 1 (Krause & Golovin Defs. 1.1–1.3)**.** For $f : 2^V \to \mathbb{R}$, the discrete derivative is $\Delta(e \mid S) := f(S \cup \{e\}) - f(S)$. f is *submodular* if for every $A \subseteq B \subseteq V$ and $e \in V \setminus B$, $\Delta(e \mid A) \geq \Delta(e \mid B)$; equivalently, for all $A, B \subseteq V$, $f(A \cap B) + f(A \cup B) \leq f(A) + f(B)$. f is *monotone* if $f(A) \leq f(B)$ whenever $A \subseteq B$, i.e. every discrete derivative is nonnegative.

Krause & Golovin's running example (their Figure 1) is placing sensors in a water network: a new sensor adds less newly covered area when more sensors are already placed. Replace "sensor" with "inspection." Monotone submodular examples include weighted coverage, facility location $f(S) = \sum_i \max_{j \in S} M_{ij}$ with nonnegative M, entropy, and mutual information with conditionally independent observations. Submodularity is preserved under nonnegative combinations and truncation $\min\{f, c\}$ (Section 1.2).

A consequence worth naming: inspecting (or fixing) the same item again has $\Delta(e \mid S) = 0$ when $e \in S$. An agent that loops on the same fix is adding items of zero marginal value — the submodular view of why looping happens, in the syllabus's phrase.

4.3 Slide 25 – Greedy and the $(1 - 1/e)$ Guarantee

The greedy algorithm starts from $S_0 = \emptyset$ and sets $S_i = S_{i-1} \cup \{\arg\max_e \Delta(e \mid S_{i-1})\}$.

Theorem 1 (Nemhauser, Wolsey & Fisher 1978; Krause & Golovin Thm. 1.5)**.** Fix a nonnegative monotone submodular f and let $\{S_i\}$ be the greedy sets. For all positive integers k and ℓ, $f(S_\ell) \geq (1 - e^{-\ell/k}) \max_{|S| \leq k} f(S)$. In particular $f(S_k) \geq (1 - 1/e)\,\mathrm{OPT}_k \approx 0.632\,\mathrm{OPT}_k$.

Proof sketch. Let S^* be optimal of size k. For each $i < \ell$: $f(S^*) \leq f(S^* \cup S_i)$ by monotonicity; this equals $f(S_i)$ plus a telescoping sum of marginal gains of the elements of S^* added one by one; each such gain is at most $\Delta(v \mid S_i)$ by submodularity; each $\Delta(v \mid S_i)$ is at most $f(S_{i+1}) - f(S_i)$ by the greedy choice; and there are at most k terms. Hence $f(S^*) - f(S_i) \leq k\,[f(S_{i+1}) - f(S_i)]$. Writing $\delta_i = f(S^*) - f(S_i)$ gives $\delta_{i+1} \leq (1 - 1/k)\delta_i$, so $\delta_\ell \leq (1 - 1/k)^\ell \delta_0 \leq e^{-\ell/k} f(S^*)$, using $\delta_0 \leq f(S^*)$ and $1 - x \leq e^{-x}$. □

The bound is tight: no algorithm evaluating f at polynomially many sets can guarantee better than $1 - 1/e$ (Nemhauser & Wolsey 1978), and $(1 - 1/e)$ is NP-hard to beat even for maximum coverage (Feige 1998). Over-budget helps: $\ell = 5k$ improves the ratio from about 0.63 to about 0.99. Lazy evaluations (Minoux 1978) exploit the fact that marginal gains only decrease, so stale upper bounds prune most evaluations. PA5 Task 4 asks you to run greedy on f, plot coverage against k next to random selection and the $(1 - 1/e)f(V)$ reference line (the bound at $k = n$; for $k < n$ the guarantee is relative to OPT_k, which you cannot compute), and verify that the marginal gains are nonincreasing. The theorem is classical; its application to AI-control monitoring is, so far, an experiment you run in PA5 rather than a result you can cite.

4.4 Slide 26 – Regime 2: Adaptive Submodularity

In Regime 2, inspections reveal information: an audit's outcome should change where you look next. The object to optimize is a *policy* π mapping partial realizations ψ (items selected so far and what each revealed) to the next item. Following Krause & Golovin Section 5, the world state is a random variable Φ with a known prior, $\phi(v)$ is the outcome of inspecting v under realization ϕ, and $f(A, \phi)$ is the reward of having inspected A in state ϕ.

Definition 2 (Krause & Golovin Defs. 1.11–1.13)**.** The conditional expected marginal benefit is $\Delta(v \mid \psi) := \mathbb{E}[f(\mathrm{dom}(\psi) \cup \{v\}, \Phi) - f(\mathrm{dom}(\psi), \Phi) \mid \psi]$. f is *adaptive monotone* if $\Delta(v \mid \psi) \geq 0$ for every ψ of positive probability, and *adaptive submodular* if $\Delta(v \mid \psi) \geq \Delta(v \mid \psi')$ whenever ψ is a sub-realization of ψ' and $v \notin \mathrm{dom}(\psi')$.

Adaptive greedy repeatedly selects $\arg\max_v \Delta(v \mid \psi)$, observes $\phi(v)$, and updates ψ. Theorem 1.14 (Golovin & Krause 2011) gives $f_{\mathrm{avg}}(\pi_\ell^{\mathrm{greedy}}) \geq (1 - e^{-\ell/k}) f_{\mathrm{avg}}(\pi_k^*)$ for any policy π_k^* selecting at most k items — the same guarantee over policies.

Part II's protocols already adapt: shutting down after one caught backdoor, switching to T when false positives run high (Greenblatt et al. Appendix G). The central caveat: Φ is a fixed sample from a known prior and does not change in response to inspections. Removing that assumption is Regime 3.

Remark 2 (Open question: adaptive inspection of agents)**.** No published work applies adaptive submodularity to sequential inspection of AI agents. The framework fits: inspecting one action reveals information (clean, suspicious, backdoored) that changes which action to inspect next, which is exactly the policy-over-partial-realizations setting of Golovin & Krause. The nearest instance is Akcin et al. (CoRL 2024), who run adaptive submodular greedy to allocate human supervisors across a robot fleet, where the overseen systems are robots collecting imitation-learning data rather than language agents under adversarial pressure. Whether the adaptive greedy guarantee improves on a static threshold policy such as Greenblatt et al.'s trusted monitoring — in safety, in usefulness, or in audit budget — is an open empirical and theoretical question. Note also that adaptive submodularity of the PA5 coverage objective would have to be checked, not assumed: Theorem 1.14 requires it.

4.5 Slide 27 – Regime 3: When the Inspected Agent Is Strategic

Independence breaks. A scheming adversary concentrates attacks where monitoring is weak (PA5 handout, Hint 5; Griffin et al. Sections 3 and 8). Then the probability p_j that item j is attacked depends on the inspection set S, and the fixed-p_j coverage model is wrong. (Robust submodular optimization and the SATURATE algorithm are treated in Lecture 2, alongside adversarial submodular games.)

Randomize and commit. Choose a distribution p over inspection sets of size at most k and maximize the worst-case expected value $U(p) = \min_i \mathbb{E}_{S \sim p}[f_i(S)]$ against an adversary who knows p. Krause, Roper & Golovin (2011) give an efficiently computable $\hat{p}$ with $U(\hat{p}) \geq (1 - 1/e) U(p^*) - \varepsilon$. This is the Stackelberg structure of the AI-Control Game (Blue commits, Red best-responds) and matches Griffin et al.'s finding that mixed protocols fill the weak-monitor frontier.

Credible threats. Shutdown-on-detection is a trigger strategy: a repeated-game device where the first detected deviation triggers a permanent punishment — here, shutdown (Greenblatt et al. Table 1). Under trusted monitoring the probability that an attempt is audited is 0.338 and sneaky backdoors rarely work (Section 3.2.1), so the red team's best response is to attack rarely and subtly. But Red is memoryless in Griffin et al.'s model (Section 3), so this is not a repeated game with punishment phases on both sides. The deterrence comes from Blue's ex-ante commitment: the commitment to shut down on detection, not the inspection count, does the work — the leader–follower logic of the AI-Control Game (Slide 19), with the agent as the follower.

4.6 Slide 28 – Research Frontier: Who Has Used Submodularity for Oversight?

This slide reports a literature survey completed on 2 November 2026 that searched for published uses of submodular optimization in allocating oversight to AI agents. The connection is real but thin. Four anchors are the nearest published work; none is inside the AI-control literature.

- **Akcin, Tanriverdi, Kale, and Chinchali (CoRL 2024).** "Fleet Supervisor Allocation: A Submodular Maximization Approach." Human supervisors are allocated across a fleet of robots by an adaptive submodular allocation policy under uncertain network connectivity. This is the one peer-reviewed instance of submodular oversight allocation the survey found; the overseen systems are robots, not language agents, and the objective is the informativeness and diversity of collected data rather than safety.
- **Itkin (2026a, 2026b).** Two single-author preprints (arXiv:2606.27409 and arXiv:2607.09765) place verifier or corrector agents in an LLM multi-agent network modeled as consensus on a graph. The first derives a supermodular placement objective with a greedy $(1 - 1/e)$ rule for a limited corrector budget; the second shows that a coherence objective stays submodular under heterogeneous corrector strength, so a cost–benefit greedy is within $1 - 1/e$ of the best placement at any budget. Closest in spirit to Part III, but preprints, and the agents being corrected are unreliable rather than adversarial.
- **Hua, Baskerville, Lemoine, Hopman, Bhatt, and Tracy (NeurIPS 2025).** "Combining Cost-Constrained Runtime Monitors for AI Safety." Several monitors with different costs and accuracies must be combined under an average-case budget to maximize recall of misaligned outputs. This is the natural slot for a submodular treatment; the paper instead searches exhaustively over when and which monitors to call and allocates interventions by the Neyman–Pearson lemma. Whether the objective is submodular in the set of monitors called is not asked.
- **Wang, Lin, Lou, and Li (2026).** "BRA-Audit" (arXiv:2608.14668) schedules audit calls in an LLM multi-agent system as audit-point placement under a fixed audit-call budget, minimizing cumulative unchecked exposure on a dynamic dependency graph. The scheduler is greedy, prioritizing influential and long-unaudited regions, but the paper makes no submodularity claim and gives no approximation guarantee. Open question: is the cumulative-exposure objective submodular? If it is, the greedy scheduler inherits a $(1 - 1/e)$ bound.

What is missing is any paper in the AI-control literature proper. The survey found zero occurrences of "submodular" in Greenblatt et al. (2024), Griffin et al. (2024), the ControlArena codebase, or the public output of any AI safety lab, and no paper applying Golovin & Krause's adaptive greedy to sequential inspection of AI agents. The observation on Slide 23 that the coverage function is monotone submodular appears to be original to this course. The honest framing for Part III is therefore: the structure fits, the theorems are off the shelf, and the application is unwritten. PA5 Task 4 is a first experiment, not a reproduction of a published result. For a student looking for a project, two concrete openings are (i) checking submodularity of the BRA-Audit exposure objective and (ii) comparing adaptive greedy against a static-threshold protocol in a ControlArena setting.

5 Part IV: PA5 and Where This Goes

5.1 Slide 30 – PA5 Preview: Reproduce AI Control Monitoring

Programming Assignment 5 (due Week 14) has five tasks.

1. **Environment (10 points).** Thirty coding tasks with tests, a subprocess sandbox with a timeout, and three scripted backdoor patterns that pass the tests but fail an edge case.
2. **Agent and monitor (25 points).** An untrusted agent in honest and adversarial modes, and a trusted monitor that sees the task and code (not the tests) and returns a suspicion score $s \in [0, 1]$; accept if $s \leq \tau$.

3. **Safety–usefulness frontier (25 points).** Sweep 21 thresholds; safety is the fraction of adversarial outputs flagged, usefulness the fraction of honest outputs accepted; add bootstrap bands and the no-, full-, and random-monitoring baselines.
4. **Submodular budget analysis (20 points).** Define $f(S) = 1 - \prod_{j \in S}(1 - p_j d_j)$, run greedy for $k = 1, \ldots, 30$ against random subsets and the $(1 - 1/e)f(V)$ reference line (the bound at $k = n$; for $k < n$ the guarantee is relative to OPT_k, which you cannot compute), verify nonincreasing marginal gains, and discuss a concentrating adversary.
5. **Analysis (20 points).** Monitor quality, comparison to Greenblatt et al., submodularity in practice, and the PA1–PA5 arc.

About 120 API calls are needed, the lightest of any PA. You reuse PA1's estimation discipline and PA4's protocol machinery, produce Greenblatt et al.'s Figure 2 in miniature, and ask Part III's question of your own data; current instructions are on Canvas.

5.2 Slide 31 – ControlArena: The Research Harness (Optional for PA5)

ControlArena is a Python library from the UK AI Security Institute and Redwood Research for AI-control experiments. *Settings* pair a benign main task with a harmful side task; *policies* are honest, attacking, trusted, or untrusted model organisms; *monitors* produce suspicion scores; *micro-protocols* such as defer-to-trusted act within a trajectory; *macro-protocols* cover auditing and shutdown; scorers report safety and usefulness. Settings span code sabotage, infrastructure, data poisoning with collusion-propensity side tasks, evaluation sabotage, prompt injection, and real-world agent tasks. The README asks users to keep the tasks out of training data and out of any API that may train on inputs (Inglis et al. 2025).

5.3 Slide 32 – How Today Connects, and What Comes Next

- **Module 7 to Module 8.** The channel cuts both ways, so monitor the channel; collusion between policy and monitor is the same phenomenon inside the oversight stack.
- **Module 4 (Stackelberg) and Module 7 (double oracle) to Module 8.** Commitment (leader–follower) and the double oracle solve the control game. Module 3's no-regret view reappears as online submodular maximization: an online greedy algorithm whose average $(1 - 1/e)$-regret against the best fixed set of size k vanishes over rounds (Krause & Golovin, Section 4) — a monitor that learns which items to inspect.
- **Module 8, Lecture 2:** adversarial submodular games (IGG), budget-limited intervention, and the poison-pill hardness construction. **Module 9:** poker, Cicero, agent benchmarks, and evaluation methodology — where the systems we would monitor come from.

Three sentences to keep. Monitoring is a visibility problem: who, what, and how. Scaling it is a game, because the watched model knows. Spending the budget is an optimization problem with a guarantee — but only after checking the structure of what you are maximizing.

References

- Chan, A., Ezell, C., Kaufmann, M., Wei, K., Hammond, L., Bradley, H., Bluemke, E., Rajkumar, N., Krueger, D., Kolt, N., Heim, L., and Anderljung, M. (2024). "Visibility into AI Agents." ACM FAccT 2024; arXiv:2401.13138.

- Baker, B., Huizinga, J., Gao, L., Dou, Z., Guan, M. Y., Madry, A., Zaremba, W., Pachocki, J., and Farhi, D. (2025). "Monitoring Reasoning Models for Misbehavior and the Risks of Promoting Obfuscation." arXiv:2503.11926.
- Greenblatt, R., Shlegeris, B., Sachan, K., and Roger, F. (2024). "AI Control: Improving Safety Despite Intentional Subversion." ICML 2024, PMLR 235: 16295–16336; arXiv:2312.06942. The PA5 reproduction target.
- Griffin, C., Thomson, L., Shlegeris, B., and Abate, A. (2024). "Games for AI Control: Models of Safety Evaluations of AI Deployment Protocols." arXiv:2409.07985; presented at IASEAI 2026.
- Krause, A., and Golovin, D. (2014). "Submodular Function Maximization." In *Tractability: Practical Approaches to Hard Problems*, Chapter 3, Cambridge University Press.
- Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. (1978). "An Analysis of Approximations for Maximizing Submodular Set Functions—I." *Mathematical Programming* 14(1): 265–294.
- Nemhauser, G. L., and Wolsey, L. A. (1978). "Best Algorithms for Approximating the Maximum of a Submodular Set Function." *Mathematics of Operations Research* 3(3): 177–188.
- Golovin, D., and Krause, A. (2011). "Adaptive Submodularity: Theory and Applications in Active Learning and Stochastic Optimization." *Journal of Artificial Intelligence Research* 42: 427–486.
- Inglis, R., Matthews, O., Tracy, T., Makins, O., et al. (2025). "ControlArena." UK AI Security Institute and Redwood Research; README (course copy in the Module 8 references folder).
- Perez, E., Huang, S., Song, F., Cai, T., Ring, R., Aslanides, J., Glaese, A., McAleese, N., and Irving, G. (2022). "Red Teaming Language Models with Language Models." EMNLP 2022, 3419–3448. Syllabus reading.
- Feige, U. (1998). "A Threshold of $\ln n$ for Approximating Set Cover." *J. ACM* 45(4): 634–652. Minoux, M. (1978). "Accelerated Greedy Algorithms for Maximizing Submodular Set Functions." *Optimization Techniques*, LNCS, 234–243.
- Krause, A., McMahan, B., Guestrin, C., and Gupta, A. (2008). "Robust Submodular Observation Selection." *JMLR* 9: 2761–2801. Krause, A., Roper, A., and Golovin, D. (2011). "Randomized Sensing in Adversarial Environments." IJCAI 2011.
- Avenhaus, R., von Stengel, B., and Zamir, S. (2002). "Inspection Games." *Handbook of Game Theory*, Vol. 3, Ch. 51. Dresher, M. (1962). "A Sampling Inspection Problem in Arms Control Agreements." RAND RM-2972.
- Bowman, S. R., et al. (2022). "Measuring Progress on Scalable Oversight for Large Language Models." arXiv:2211.03540. Lecture 7.1 sources: OpenAI (2026) incident technical report; Kitaoka et al. (2026), arXiv:2610.24052; Hoffmann (2026), Zenodo.
- Akcin, O., Tanriverdi, A. E., Kale, K., and Chinchali, S. P. (2024). "Fleet Supervisor Allocation: A Submodular Maximization Approach." *CoRL 2024*, PMLR 270: 4603–4630.
- Itkin, I. (2026a). "Delayed Verification Destabilizes Multi-Agent LLM Belief: Instability Thresholds and Optimal Corrector Placement." arXiv:2606.27409 (preprint). Itkin, I. (2026b). "How Much Does Correctness Cost? Budgeted Placement of Strong Correctors in a Weak Multi-Agent Swarm." arXiv:2607.09765 (preprint).
- Hua, T. T., Baskerville, J., Lemoine, H., Hopman, M., Bhatt, A., and Tracy, T. (2025). "Combining Cost-Constrained Runtime Monitors for AI Safety." *NeurIPS 2025*; arXiv:2507.15886.
- Wang, K., Lin, Y., Lou, J., and Li, J. (2026). "BRA-Audit: Budgeted Runtime Auditing for LLM Multi-Agent Systems via Cumulative-Exposure Audit-Point Placement." arXiv:2608.14668 (preprint).