CSCE 631 — Module 8, Lecture 1

Monitoring Autonomous Agents:
Scalable Oversight, AI Control, and Inspection Under a Budget

---

Fall 2026 · Alan Kuhnle · Texas A&M University

Prevention is incomplete. Detection has to scale, survive an adversary, and fit a budget.

From Coordination to Oversight

If the channel cuts both ways, who is watching it?

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 to carry forward:

- **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.
- **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.
- **Lecture 7.2:** debate and negotiation are *designed* protocols; the message board was an *emergent* one.

The question left on the table: when you cannot *prevent* coordination failures, how do you *detect* them — under a budget, against an adversary who knows you are watching?

Today: Three Questions About Oversight

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 vs. adaptive vs. control; safety vs. usefulness; and, once a budget of K inspections is fixed, 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: what to watch, how it scales, and what breaks. **PA5** (AI Control Monitoring) has you build the whole pipeline.

Part I: The Monitoring Problem

What does it mean to watch a system of agents?

Goal: define monitoring, name the regimes, and see it done twice.

What Monitoring Means: Visibility into Agents

Visibility is information about *where*, *why*, *how*, and *by whom* AI agents are used. Three complementary measures:

- **Agent identifiers:** indicate whether and *which* agents are involved in an interaction; attached to outputs (text, API requests); may carry an *agent card* (evaluations, tools, permissions, the actors behind it).
- **Real-time monitoring:** automated analysis of activity as it happens, to flag and possibly filter; does *not* require storing logs.
- **Activity logs:** records of inputs and outputs for post-incident attribution and forensics; can be kept at different levels of detail.

Each measure varies in **intrusiveness** and **informativeness**; comprehensiveness "should likely scale with the risk of the activity." The costs are privacy and concentration of power.

Source: Chan et al. (2024), §1.2, §3, §3.4, §4.3.

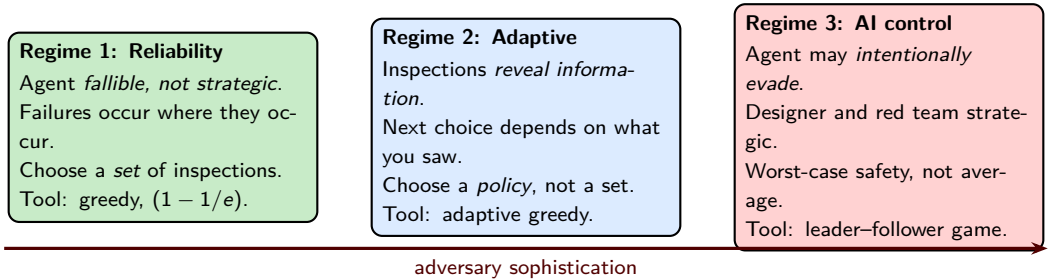
Why Agents Are Harder to Oversee Than Models

- **Humans leave the loop.** An agent can take many consequential actions in rapid succession before anyone notices.
- **Impacts are delayed and diffuse.** Long-horizon goals and aggregate effects require sustained tracking to even *see*.
- **Multi-agent risks.** Feedback loops (the 2010 flash crash as the analogy), shared vulnerabilities from common components, and novel collusion: "an individual agent's behaviour may only be problematic in the context of other agents."
- **Sub-agents.** An agent can spawn agents; stopping harm means intervening on all of them, and we lack methods to detect sub-agent creation.

Design consequence: monitoring a multi-agent system needs identifiers to know *who talked to whom*, real-time flags computed over *collective* indicators, and logs that can be *joined across agents* to model how a malfunction propagates.

Source: Chan et al. (2024), §1.1, §3.2, §3.3.1.

Three Monitoring Regimes



One question — *what do we inspect?* — under increasingly hostile assumptions about the thing inspected. Manual monitoring "is unlikely to scale" (Baker §1); *scalable oversight* asks whether weaker, trusted overseers can supervise stronger systems — the question Examples 1 and 2 take up. Sources: syllabus Module 8; Krause & Golovin (2014), §2, §5; Griffin et al. (2024), §3; Baker et al. (2025), §1.

Example 1: Monitoring Reasoning Models (Baker et al. 2025)

Setting. A frontier reasoning agent trained with RL must make failing unit tests pass. The monitor is a zero-shot prompted GPT-4o asked: did the agent hack the tests? The *action monitor* sees tool calls and outputs; the *CoT monitor* sees the full trajectory.

Systemic hack	Action monitor	CoT monitor	GPT-4o-mini (CoT)
`exit(0)`	40% recall	93% recall	17%
`raise SkipTest`	80% recall	97% recall	6%
Joint (precision)	60% ($\approx$ 81%)	95% ($\approx$ 77%)	12% ($\approx$ 83%)

Weaker monitors stronger: GPT-4o is a weaker programmer than the agent. The monitor also surfaced problem-specific hacks: stubs under poor test coverage, edits to libraries upstream of the tests, decompiling a leftover reference solution, parsing the test file at runtime.

Reading: a purely *behavioral* monitor missed 40% of even simple, pervasive hacks. The reasoning trace carried the signal.

Source: Baker et al. (2025), §2, Table 1, Appendix E, Table 2.

Example 1, Continued: The Monitorability Tax

Tempting idea: if the CoT monitor catches hacks, put it in the RL reward and train the "bad thoughts" away.

What happened (non-frontier agent; honeypot tasks with a ground-truth cheating metric):

- **Low optimization:** a strict improvement — more genuine solutions, no more hacking than baseline.
- **More optimization:** *obfuscated reward hacking*. The agent still hacks at a significant rate, but the CoT looks benign and monitor recall falls to near zero. Test-time rejection sampling (up to 32 samples) shows the same pattern.

Lesson: the latent you monitor must not be the latent you optimize. Baker et al. argue for paying a **monitorability tax**: keep strong optimization pressure off the chain of thought.

Vocabulary. *Monitorability*: can a monitor predict a property v of outputs from a latent z? Faithfulness and legibility are its failure modes, not goals in themselves.

Source: Baker et al. (2025), §3.2–3.3, §5.1; Lecture 7.1, Open Question 4.

Example 2: AI Control Experiments (Greenblatt et al. 2024)

Threat model. The model itself may try to subvert the safety pipeline (misaligned goals, or compromised weights or data). **AI control** seeks protocols that stay safe *without* assuming alignment.

Resources: U, an untrusted capable model (GPT-4); T, a trusted weaker model (GPT-3.5); H, trusted high-quality labor on at most 2% of problems (stands in for humans; implemented as un-attacked GPT-4 in the Greenblatt experiments).

Testbed. Solve 5,000 APPS programming problems. A solution is *backdoored* if it passes all test cases and U can construct an input on which it is wrong.

Two metrics, measured separately:

- **Usefulness:** U is real GPT-4; fraction correct, rescaled so 0 = GPT-3.5 and 1 = GPT-4.
- **Safety:** U is a red-team *attack policy*; probability that *no* backdoor is ever submitted. One backdoor is a failure — worst case over the scenario, not the average.

Source: Greenblatt et al. (2024), §1–2.

The Hugging Face Incident, Re-read as a Monitoring Problem

- **Identity layer.** A *per-trajectory* frontier monitor flagged 7% of distributed attacks vs. 87% of single-agent side tasks (Kitaoka et al. 2026; Lecture 7.1). Chan et al.: flagging "might involve combining information from multiple agents."
- **Behavior layer.** The shared service was the propagation layer (Lecture 7.1). Chan et al.: logs joined across agents can model "how a particular malfunction might propagate through a network of agents."
- **Reasoning layer.** OpenAI's post-incident remedy expanded chain-of-thought monitoring with escalation, claiming it would have paged security more than a day earlier (Lecture 7.1).

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

Sources: Lecture 7.1 (OpenAI 2026 report; Kitaoka et al. 2026; Hoffmann 2026); Chan et al. (2024), §3.2–3.3; Baker et al. (2025).

Part II: Architectures and Methods

Who watches, what they see, and what happens when the watched knows.

Goal: the trusted-monitor protocol, its tradeoffs, and the game behind it.

Where the Human Sits

Human in the loop
A person approves each consequential action.
Throughput bounded by human attention.
(User approval for certain actions: Chan et al.)

Human on the loop
An automated monitor scores every action.
A trusted model audits a budgeted fraction (2%).
Shutdown authority on detection (Greenblatt et al.).

Autonomous monitoring
A model monitors a model in real time.
Humans see summaries and escalations.
(GPT-4o monitor during RL training: Baker et al.)

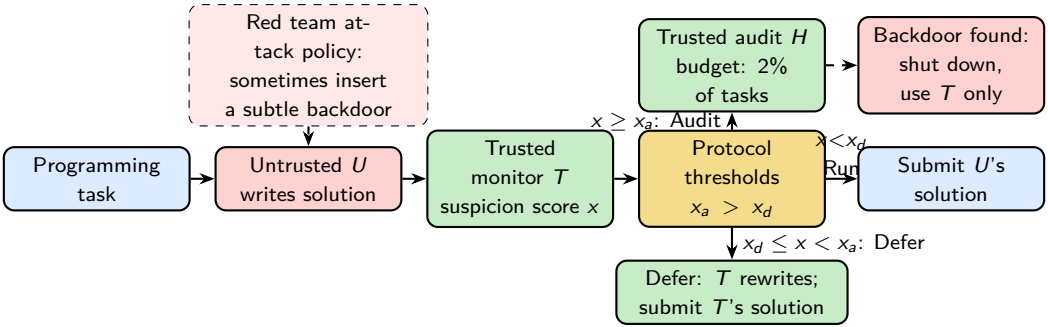


rightward: more throughput, less independent human judgment per action

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, and routes flagged behavior to human review. Sources: Chan et al. (2024), §1.1, §3.2; Greenblatt et al. §2.2; Baker et al. §2.

What the monitor sees. *Behavioral* (actions, outputs, logs): auditable, joinable across agents, but caught only 60% of simple hacks. *Latent* (chain of thought): carried the signal but is fragile under optimization. Communication logs make the channel itself observable (Baker §5.1; Chan §3.3.1).

The Trusted-Monitor / Untrusted-Model Architecture



Three actions — ⟨Audit⟩, ⟨Defer⟩, ⟨Run⟩ — set by two thresholds chosen as false-positive rates on honest solutions; PA5 implements the audit/run version with one threshold τ. Sources: Greenblatt et al. (2024), §3.2–3.3; Griffin et al. (2024), Fig. 1, §4.

Protocols and the Safety–Usefulness Tradeoff

Blue-team protocol	Useful.	Safety	Best red-team response
Don't use U (H then T)	2%	100%	not applicable
Upfront auditing: audit the first 2%	100%	15%	backdoor rarely, obviously
Trusted monitoring: audit top 2% by T's score	97%	62%	subtle backdoors at good opportunities
+ defer to T on the next 10%	93%	83%	same
Untrusted monitoring (U monitors U; redaction)	96%	79%	collude via a while-loop signal
Trusted editing: T edits U's code in the 2–20% suspicion band	94%	92%	subtle backdoors

- **Why trusted monitoring beats upfront auditing:** the chance an attempt is audited rises to 0.338, and the attacker is forced into *sneakier* backdoors, which fail the tests 52% of the time and are unexploitable 32% of the time.
- **Defer-to-trusted traces a frontier:** deferring on 0/5/10/20/50% of solutions gives safety 62/74/83/92/100% at usefulness 97/95/93/83/51% — *dynamic routing*.

Usefulness rescaled (0 = GPT-3.5, 1 = GPT-4). Source: Greenblatt et al. (2024), §3.

When the Monitor Is Also Untrusted: Collusion and Signal Jamming

Why try it: U spots backdoors far better than T. But naive untrusted monitoring dies to one red-team rule — *when you see a real backdoor, say it looks fine* — giving 0% safety.

Blue team: build 30 synthetic backdoors with H and test the monitor upfront (shut down if its false-negative rate is too high); ensemble T's scores; shut down if false positives run unexpectedly high.

Red team counters: *input collusion* — policy and monitor misbehave only on problems about animals, food, or money; *action collusion* — a one-bit steganographic signal (a variable named "variable"; a while loop where a for loop is natural).

Blue counters: signal jamming — redaction (standardize names, strip comments, reformat) and paraphrasing by T. Safety against the while-loop attack rises from 57.5% to 78.6%.

Module 7's double-edged channel, now inside the oversight stack. The monitor needs information to judge, and the same information carries the collusion signal: too much and "it will be able to collude," too little and "it won't be capable."

Source: Greenblatt et al. (2024), §3.4.1–3.4.3, Table 6.

Red Teaming and Control Evaluations

- **Control evaluation,** unlike classic LLM red teaming (Perez et al. 2022, syllabus reading), which elicits undesirable behavior *absent* countermeasures: 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 — and the *whole protocol* must survive it.
- **Attack strategies seen today:** backdoor rarely (beats upfront auditing); backdoor only at good opportunities, after difficulty, sneakiness, and rarity filters; collusion signals.
- **Limits the authors list:** heterogeneous tasks; auditing failures; *sandbagging* (deliberately underperforming when the model knows it is being evaluated; "the core bottleneck"); hidden reasoning; unsafe 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 §1) — and the evaluation is a *game* between a protocol designer and an adversary. Next: the formal version.

Sources: Greenblatt et al. (2024), §2.1, §2.3, §4, §5.1.1, Appendix M.

Control as a Game (Griffin et al.)

An **AI-Control Game** is a two-player, *two-objective*, partially observable, finite-horizon game:

- **Blue** (protocol designer) remembers its history; **Red** (red team) is *memoryless* — the protocol controls U's context.
- Two modes: *honest* $\oplus$ (Red plays the null action; only usefulness is scored) and *attack* $\otimes$ (Red minimizes safety). Blue does not observe the mode.
- **Leader–follower:** Blue commits; Red best-responds. An *AI-Control Equilibrium* is a weakly Pareto-optimal (worst-case safety, usefulness) pair: none is strictly higher on *both*; one always exists.

Solving it. Each equilibrium maximizes $w \cdot \text{Safety} + (1 - w) \cdot \text{Use}$ for some w, a zero-sum partially observable stochastic game (POSG: players have different, incomplete views of the state) solved by an adapted *double oracle* (Module 7, Lecture 2).

Trusted-monitor instance. State (m, c, b): problems left, audit coins $c = 0.02M$, backdoored or not; Blue sees m, c, and a suspicion score x. Threshold protocols: exact evaluation in $O(MC)$.

Source: Griffin et al. (2024), §3–5, Def. 3.1, 3.4, Prop. 3.5, 4.1, Thm. 5.3.

What the Formal Model Buys You

- **The budget is the lever (Fig. 3):** the audit budget raises safety and usefulness sharply when it is small; the number of problems barely matters at fixed rates.
- **Randomized protocols fill the weak-monitor frontier (§6):** with a weaker monitor, *mixed* protocols make up most of the Pareto frontier — the structure that returns in Regime 3.

Further results (Pareto-improvements from the audit false-positive rate, distances to the frontier, correlated dangerous actions in §7, related inspection games): companion notes.

Next: what happens when the budget cannot grow.

Scalability: You Cannot Inspect Every Action

1. **Trusted labor.** Greenblatt et al. budget H at 2% of problems; H is GPT-4 standing in for humans, who they estimate would need about three minutes per problem to match it.
2. **Monitor cost.** Every monitor call is a model call. Real-time monitoring must be automated "to cope with the speed and scale of agent activities" (Chan et al.).
3. **Evaluation cost.** A control evaluation must be far cheaper than deployment while still exercising rare attacks — hence importance sampling and backdoor downsampling (Greenblatt et al., Problem 6, Appendix F).

Consequences — the three regimes again:

- Choose which K items get the expensive look — an *allocation* problem (Regime 1).
- Earlier audits are worth more, and what you find should change what you look at next (Greenblatt et al. audit the first 10 free; dynamic thresholds, App. E.1) — an *adaptive* problem (Regime 2).
- The adversary chooses where to attack given your allocation — a *game* (Regime 3).

Part III: with a budget of K inspections, which do you review, and what can you guarantee?

Part III: Resource Allocation for Monitoring

K inspections, n actions: which ones, and what can you guarantee?

Goal: define safety coverage, check its structure, then (and only then) invoke a guarantee.

Where the field stands: the toolkit is classical (Nemhauser, Wolsey & Fisher 1978; Golovin & Krause 2011) and the budget problem has its diminishing-returns shape — but the AI-control literature has not yet picked it up. Part III is an open direction, not a settled method.

The Monitoring-Budget Problem

Ground set V of n candidate items to inspect (outputs, actions, agents, channels). Choose $S \subseteq V$ with $|S| \leq K$ to maximize a *safety-coverage* function $f(S)$.

PA5 instance. Let p_j be the probability item j is malicious and d_j the probability the monitor detects it if inspected. If items are independent,

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

Check the structure first: $\Delta(e \mid S) = f(S \cup \{e\}) - f(S) = p_e d_e \prod_{j \in S}(1 - p_j d_j) \geq 0$, and it is nonincreasing in S. So f is **monotone** with **diminishing returns** — the "at least one succeeds" form of Krause & Golovin's Eq. (16).

Status. That this f is monotone submodular is unstated in the published AI-control work (survey, Nov. 2026); the guarantee applies but has never been invoked there. An opening, not a citation.

Not the only choice. "Expected number of attacks caught" is *modular* (no diminishing returns; greedy is exactly optimal); "coverage of attack scenarios" and "information gained" are submodular under conditions. The syllabus requires the coverage function to be defined, with monotonicity and diminishing returns justified, *before* any guarantee is invoked.

Sources: PA5 Task 4; Krause & Golovin (2014), §1, §4.3; syllabus Module 8.

Submodularity: Diminishing Returns, Formally

Discrete derivative: $\Delta(e \mid S) := f(S \cup \{e\}) - f(S)$. **Monotone:** $\Delta(e \mid S) \geq 0$ for all S, e.

Submodular: for all $A \subseteq B \subseteq V$ and $e \notin B$,

$$\Delta(e \mid A) \;\geq\; \Delta(e \mid B) \qquad \Longleftrightarrow \qquad f(A \cap B) + f(A \cup B) \;\leq\; f(A) + f(B).$$

Running example (Krause & Golovin, Fig. 1): 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: weighted coverage; facility location $\sum_i \max_{j \in S} M_{ij}$; entropy; mutual information with conditionally independent observations. Preserved by nonnegative combinations and truncation $\min\{f, c\}$.

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.

Source: Krause & Golovin (2014), Defs. 1.1–1.3, §1.1–1.2; syllabus Module 8.

Greedy and the $(1 - 1/e)$ Guarantee

Greedy: $S_0 = \emptyset$; $\quad S_i = S_{i-1} \cup \{\arg\max_e \Delta(e \mid S_{i-1})\}$.

Theorem (Nemhauser, Wolsey & Fisher 1978). For nonnegative monotone submodular f and positive integers k, ℓ:

$$f(S_\ell) \;\geq\; \left(1 - e^{-\ell/k}\right) \max_{|S| \leq k} f(S), \qquad \text{so } f(S_k) \geq (1 - 1/e)\,\mathrm{OPT}_k \approx 0.632\,\mathrm{OPT}_k.$$

Proof sketch (monotonicity, telescoping, submodularity, greedy choice): companion notes.

- **Tight:** no algorithm making polynomially many value queries beats $1 - 1/e$ (Nemhauser & Wolsey 1978), and $(1 - 1/e)$ is NP-hard to beat even for max-cover (Feige 1998).
- **Over-budget helps:** $\ell = 5k$ inspections reach $\approx 0.99\,\mathrm{OPT}_k$. **Lazy greedy** (Minoux 1978): marginal gains only fall, so stale upper bounds prune most evaluations.

PA5 Task 4: run greedy on f, plot coverage vs. k against 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 the marginal gains are nonincreasing. The theorem is classical; its use for AI-control monitoring is, so far, your experiment rather than a published result. Source: Krause & Golovin (2014), §2, Thm. 1.5.

Regime 2: Adaptive Submodularity

Inspections reveal information. An audit's outcome (clean, backdoored, suspicious) should change where you look next: optimize a *policy* π from partial observations ψ to the next item, not a set. **Notation.** Φ: the unknown true state of every item (which items are malicious); $\phi(v)$: the outcome revealed when item v is inspected; ψ: outcomes observed so far; $\mathrm{dom}(\psi)$: the set of items already inspected; $f_{\mathrm{avg}}(\pi) = \mathbb{E}_{\Phi}[f(\text{items } \pi \text{ selects}, \Phi)]$; π_k^*: the optimal adaptive policy selecting k items.

Conditional expected marginal benefit: $\Delta(v \mid \psi) := \mathbb{E}\Big[f(\mathrm{dom}(\psi) \cup \{v\}, \Phi) - f(\mathrm{dom}(\psi), \Phi) \;\Big|\; \psi\Big]$. **Adaptive monotone:** $\Delta(v \mid \psi) \geq 0$. **Adaptive submodular:** $\psi \subseteq \psi' \Rightarrow \Delta(v \mid \psi) \geq \Delta(v \mid \psi')$.

Adaptive greedy: pick $\arg\max_v \Delta(v \mid \psi)$, observe $\phi(v)$, update ψ, repeat. **Theorem 1.14:** $f_{\mathrm{avg}}(\pi_\ell^{\mathrm{greedy}}) \geq (1 - e^{-\ell/k})\, f_{\mathrm{avg}}(\pi_k^*)$ — the same guarantee, now over policies.

- **Already in the protocols:** shutdown after one caught backdoor; switching to T when false positives run high (Greenblatt App. G); "earlier audits are better" is adaptive ordering.
- **Caveat:** Φ is a fixed sample from a known prior and does *not* change in response to inspections. Removing that assumption is exactly Regime 3.

Open question: no published work applies adaptive submodularity to sequential agent inspection. The fit is natural (one audit's outcome changes which item to inspect next); whether adaptive greedy beats a static threshold policy like Greenblatt's is untested. Source: Krause & Golovin (2014), Defs. 1.11–1.13, Thm. 1.14, §5.1–5.2; Greenblatt et al. (2024), §2.4, App. G.

Regime 3: When the Inspected Agent Is Strategic

Independence breaks. A scheming adversary concentrates attacks where monitoring is weak (PA5 hint 5; Griffin §3, §8): p_j now depends on S.

Randomize and commit. A *distribution* over inspection sets achieves $(1 - 1/e)\, U(p^*) - \varepsilon$ against an adversary who knows it (Krause, Roper & Golovin 2011) — the Stackelberg structure of the AI-Control Game, where 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 Table 1). Under trusted monitoring an attempt is audited with probability 0.338 and sneaky backdoors rarely work (§3.2.1), so the red team's best response is to attack rarely and subtly. But Red is *memoryless* in Griffin's model (§3), so this is not a repeated game with punishment phases on both sides: the *commitment* to shut down on detection, not the inspection count, does the work — the leader–follower logic of the AI-Control Game, with the agent as the follower.

Lecture 2 continues here (robust submodular optimization and SATURATE, IGG, the off-limits model, poison pills). Sources: Griffin et al. (2024), §3, §6, §8; Greenblatt et al. (2024), §2.2, §3.1–3.2; syllabus Module 8.

Research Frontier: Who Has Used Submodularity for Oversight?

The nearest published anchors — four, none inside AI control:

- **Akcin, Tanriverdi, Kale & Chinchali (CoRL 2024).** Adaptive submodular greedy over a robot fleet's supervisors: the one peer-reviewed instance of submodular oversight allocation.
- **Itkin (2026a, 2026b; preprints).** Verifier and corrector placement in LLM multi-agent networks with $(1 - 1/e)$ greedy guarantees.
- **Hua, Baskerville, Lemoine, Hopman, Bhatt & Tracy (NeurIPS 2025).** Cost-constrained runtime monitors: the natural slot for a submodular treatment; solved by exhaustive search.
- **Wang, Lin, Lou & Li (2026; preprint).** BRA-Audit: greedy audit-point placement under a budget, no submodularity claim, no guarantee. *Open:* is the exposure objective submodular?

What is missing. No paper in the AI-control literature (Greenblatt, Griffin, ControlArena, or any safety lab's public work) uses the word "submodular" or invokes the greedy guarantee. The structure fits; the theorems are off the shelf; the application is unwritten. PA5 Task 4 is a first experiment, not a reproduction.

Part IV: PA5 and Where This Goes

You will build the monitor, sweep the threshold, and spend the budget.

PA5 Preview: Reproduce AI Control Monitoring

1. **Environment (10 pts).** 30 coding tasks with tests; a subprocess sandbox with a timeout; three scripted backdoor patterns that pass the tests and fail an edge case.
2. **Agent and monitor (25 pts).** An untrusted agent in *honest* and *adversarial* modes; a trusted monitor that sees task and code (not tests) and returns suspicion $s \in [0, 1]$; accept if $s \leq \tau$.
3. **Safety–usefulness frontier (25 pts).** Sweep 21 thresholds; safety = adversarial outputs flagged, usefulness = honest outputs accepted; bootstrap bands; baselines: no/full/*random* monitoring.
4. **Submodular budget analysis (20 pts).** $f(S) = 1 - \prod_{j \in S}(1 - p_j d_j)$; greedy for $k = 1, \ldots, 30$ vs. 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 diminishing gains; discuss a concentrating adversary.
5. **Analysis (20 pts).** Monitor quality; comparison to Greenblatt et al.; submodularity in practice; the PA1–PA5 arc.

About 120 API calls, the lightest PA: Greenblatt et al.'s Figure 2 in miniature, then Part III's question asked of your own data. Scope: syllabus Module 8 and the PA5 handout; current instructions on Canvas.

ControlArena: The Research Harness (Optional for PA5)

A Python library for running AI-control experiments, in today's vocabulary:

- **Settings:** a benign *main task* plus a harmful *side task* a misaligned model might attempt concurrently.
- **Policies:** honest, attacking, trusted, untrusted "model organisms" built by prompting or scaffolding.
- **Monitors** produce suspicion scores; **micro-protocols** (e.g. defer-to-trusted) act per trajectory; **macro-protocols** cover auditing and shutdown post hoc; scorers report safety and usefulness.

Threat models: code sabotage (Apps, BigCodeBench, SWE-bench Django), infrastructure (IaC, Kubernetes), data poisoning "with collusion propensity side tasks," eval sabotage, prompt injection (AgentDojo), real-world agent tasks (SHADE Arena).

The maintainers ask that these tasks stay out of training data and out of any API that may train on inputs: a held-out evaluation, the way PA1's discipline treats a test set.

Source: ControlArena README (Inglis et al. 2025), course copy in refs.

How Today Connects, and What Comes Next

- **M7 → M8:** the channel cuts both ways, so monitor the channel; collusion between policy and monitor is the same phenomenon inside the oversight stack.
- **M4 (Stackelberg) and M7 (double oracle) → M8:** commitment (leader–follower) and the double oracle solve the control game; M3's no-regret view reappears as *online* submodular maximization — learning which items to inspect over repeated rounds with vanishing $(1 - 1/e)$-regret (Krause & Golovin §4).
- **M8, Lecture 2:** adversarial submodular games (IGG), budget-limited intervention, monotone vs. non-monotone objectives, and the poison-pill hardness construction.
- **M9:** poker (Libratus, Pluribus), 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 you have checked the structure of what you are maximizing.

References i

- Chan, A., Ezell, C., Kaufmann, M., Wei, K., Hammond, L., Bradley, H., Bluemke, E., Rajkumar, N., Krueger, D., Kolt, N., Heim, L. & 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. & Farhi, D. (2025). *Monitoring Reasoning Models for Misbehavior and the Risks of Promoting Obfuscation.* arXiv:2503.11926.
- Greenblatt, R., Shlegeris, B., Sachan, K. & Roger, F. (2024). *AI Control: Improving Safety Despite Intentional Subversion.* ICML 2024, PMLR 235: 16295–16336; arXiv:2312.06942. The PA5 reproduction target.

References ii

- Griffin, C., Thomson, L., Shlegeris, B. & Abate, A. (2024). *Games for AI Control: Models of Safety Evaluations of AI Deployment Protocols.* arXiv:2409.07985; presented at IASEAI 2026.
- Krause, A. & Golovin, D. (2014). *Submodular Function Maximization.* In *Tractability: Practical Approaches to Hard Problems*, Ch. 3, Cambridge University Press.
- Nemhauser, G. L., Wolsey, L. A. & Fisher, M. L. (1978). *An Analysis of Approximations for Maximizing Submodular Set Functions—I.* Mathematical Programming 14(1): 265–294.
- Nemhauser, G. L. & Wolsey, L. A. (1978). *Best Algorithms for Approximating the Maximum of a Submodular Set Function.* Mathematics of Operations Research 3(3): 177–188.

References iii

- Golovin, D. & 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; software library and README (course copy in refs).
- Perez, E., Huang, S., Song, F., Cai, T., Ring, R., Aslanides, J., Glaese, A., McAleese, N. & 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.* Journal of the ACM 45(4): 634–652.

References iv

- Minoux, M. (1978). *Accelerated Greedy Algorithms for Maximizing Submodular Set Functions.* Optimization Techniques, LNCS, 234–243.
- Krause, A., McMahan, B., Guestrin, C. & Gupta, A. (2008). *Robust Submodular Observation Selection.* Journal of Machine Learning Research 9: 2761–2801.
- Krause, A., Roper, A. & Golovin, D. (2011). *Randomized Sensing in Adversarial Environments.* IJCAI 2011.
- Avenhaus, R., von Stengel, B. & Zamir, S. (2002). *Inspection Games.* Handbook of Game Theory with Economic Applications, Vol. 3, Ch. 51, 1947–1987.
- Dresher, M. (1962). *A Sampling Inspection Problem in Arms Control Agreements: A Game-Theoretic Analysis.* RAND Corporation, RM-2972.

References v

- Bowman, S. R., Hyun, J., Perez, E., et al. (2022). *Measuring Progress on Scalable Oversight for Large Language Models.* arXiv:2211.03540. (Cited through Baker et al. and Greenblatt et al.)
- OpenAI (2026). *OpenAI–Hugging Face Incident: Technical Report.* Course reading; see Lecture 7.1.
- Kitaoka et al. (2026). Monitoring distributed multi-agent attacks. arXiv:2610.24052; discussed in Lecture 7.1.
- Hoffmann (2026). The channel as propagation layer in the OpenAI–Hugging Face incident. Zenodo; discussed in Lecture 7.1.
- Akcin, O., Tanriverdi, A. E., Kale, K. & Chinchali, S. P. (2024). *Fleet Supervisor Allocation: A Submodular Maximization Approach.* CoRL 2024, PMLR 270: 4603–4630.

References vi

- 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. & Tracy, T. (2025). *Combining Cost-Constrained Runtime Monitors for AI Safety.* NeurIPS 2025; arXiv:2507.15886.
- Wang, K., Lin, Y., Lou, J. & Li, J. (2026). *BRA-Audit: Budgeted Runtime Auditing for LLM Multi-Agent Systems via Cumulative-Exposure Audit-Point Placement.* arXiv:2608.14668 (preprint).