Solutions to Exam Questions 7-18

Question 7: Water Tank Problem

(a) State Space of the Abstraction

The abstraction must track water level ranges and valve state history (since the valve must hold position for 2 seconds).

State Space Definition:

- **Level states (L):** {low (0-5L), medium-low (5-20L), medium-high (20-45L), high (45-50L)}
- **Valve history:**
  - q_0: initial state
  - q_1: valve opened for 1 step
  - q_2: valve closed for 1 step
  - q_3: valve in stable state (≥2 steps)

This creates a product abstraction: States = $\{q_0 ... q_3\} \times L = 4 \times 4 =$ **16 states**

(b) LTL Safety Specification

$$\phi_s = G(\text{level} > 5) \wedge G(\text{level} < 45) \wedge$$
$$G((\text{open} \wedge X\,\text{close}) \rightarrow XX\,\text{close} \wedge XXX\text{close}) \wedge$$
$$G((\text{close} \wedge X\,\text{open}) \rightarrow XX\,\text{open} \wedge XXX\text{open})$$

Explanation:

- `G(level > 5)`: Level never drops to unsafe low
- `G(level < 45)`: Level never exceeds unsafe high
- Third clause: If valve switches from open to closed, must stay closed for 2+ steps
- Fourth clause: If valve switches from closed to open, must stay open for 2+ steps

(c) Shield Actions at Level = 8L (Valve Open 1 Second)

Current State: level=8L, valve open for 1 second → abstraction state (q_1, medium-low)

Analysis:

- Outflow: 0-2 L/sec
- If we continue open: 1 sec later, level could be 8 + 3 - 0 = 11L (worst: 3-2=1L gain)
- If we close now: valve breaks (must stay open ≥2 steps) → **FORBIDDEN by spec**

Shield Decision:

- **Allow:** open (forced to continue)
- **Forbid:** close (violates ϕ_s)

At level 8L, the shield must keep valve open for ≥1 more step. Only when level reaches high (>45L) can the shield forbid opening.

Question 8: Game Synthesis Problem

Explanation of Safety Game Construction

What the paper does (from Section 6):

1. **Product Game State Space:** $G = Q \times Q_M$
   - Q = states of specification automaton (tracking if safe)
   - Q_M = states of environment abstraction
   - Each game state is a pair (q_spec, q_env)
2. **Winning Region W:**
   - $W \subseteq F_g$ (subset of safe states)
   - Computed by backward reachability: a state is winning if:
     - It's in $F \times Q_M$ (spec is satisfied), OR
     - From this state, for ANY environment input (observation), there EXISTS a system action (shield action) that leads to another winning state
   - A state is NOT winning if the environment can force entry to unsafe states
3. **What "Winning States" Mean:**

A state is in W if the system (shield) can guarantee specification satisfaction from that point onward, regardless of environment behavior (within the abstraction's constraints).

Why shields use only winning states:

The shield's output function: $\lambda_S(g, l) = \{a \in A \mid \delta(g, l, a) \in W\}$

This outputs only actions that lead to winning states. If an action would move to a non-winning state, the shield forbids it. This ensures:

- The shield can always find a safe action (doesn't deadlock)
- Any action allowed is provably safe

Example from Water Tank:

In Figure 8 of the paper, state (q_3, q_d) is NOT winning because:

- If shield chooses "close", environment could keep level at 0-1L range
- From there, no action prevents violating the dry-tank spec
- So "close" action is forbidden, even though it might seem locally safe

---

Question 9: Correctness Proof Sketch

Proof Argument

Theorem: Any trace the shield and learner produce satisfies the specification.

Proof Structure:

1. **Shield Construction Property:** The shield is initialized at (q_0, q_0, M) and only allows transitions where both:
   - q moves according to spec automaton
   - q_M moves according to abstraction automaton
   - New state (q', q_M') ∈ W (winning)
2. **Inductively:** At each step t:
   - Current state: (q_t, q_M,t) ∈ W by construction
   - Environment produces label l_t ∈ L
   - Shield selects action a_t ∈ λ_S(q_t, l_t)
   - By definition of λ_S: next state (q_t+1, q_M,t+1) = δ(q_t, l_t, a_t) ∈ W
   - Therefore q_t+1 ∈ F (remains in specification's safe states)
3. **By induction:** All states visited satisfy q_t ∈ F throughout execution
4. **Conclusion:** The specification automaton never enters unsafe states (F is violated only if we leave F), so the trace satisfies the safety specification.

Role of F_g (Safe Game States):

F_g = (F × Q_M) ∪ (Q × (Q_M \ F_M))

This includes:

- States where spec is satisfied: F × Q_M
- States where abstraction violated: Q × (Q_M \ F_M) [these count as "wins" for shield because if abstraction fails, system isn't responsible]

This definition ensures the winning strategy only requires maintaining F, not fixing broken abstractions.

Question 10: Minimal Interference Argument

The Proof Strategy

Claim: If the shield forbids action a_k at state (q_S, q_M), then a_k must be forbidden (otherwise violations become possible).

Proof by Contradiction:

1. **Construction of Counterexample MDP M':**
   - States: S' = Q_M × L (abstraction states paired with level labels)
   - Transition: P'((q, l), a) = uniform distribution over all reachable states in abstraction that are still safe
   - This M' is consistent with abstraction ϕ_M (all traces possible in abstraction have non-zero probability in M')
2. **Why Action is Forbidden:**
   - Shield forbids a_k at (q_S, q_M) because: δ(q_S, l, a_k) ∉ W
   - Non-winning state means: adversarial environment can force spec violation

    - In the game, environment had a strategy to violate ϕ_S using abstraction-consistent traces
3. **Transfer to Real World:**
    - Since M' mirrors all possible abstract behaviors, and the environment player had a winning strategy in the game
    - If we allow a_k in M', there's non-zero probability of violating spec
    - We cannot prevent it no matter what the learner does afterward
4. **Conclusion:** Forbidding a_k is necessary. Any shield allowing it couldn't be correct.

Why This Proves Minimality:

The shield forbids only actions where there exists at least one MDP model matching the abstraction where violations are possible. For any action the shield allows, it can prove no such MDP exists, so the action is safe. Therefore, minimal interference is achieved.

Question 11: Computational Complexity Comparison

Shielding Complexity

Shield Synthesis Complexity:

- Game state space: $|Q| \times |Q_M|$ (spec states × abstraction states)
- Game solving (standard algorithm): $O(|G| \times |\Sigma_I| \times |\Sigma_O|) = O(|Q| \times |Q_M| \times |L| \times |A|)$
- Key: Depends ONLY on specification and abstraction complexity

Traditional Correct-by-Construction Complexity

Traditional Synthesis:

- Must model entire system dynamics: $|S|$ (full MDP state space)
- Often infinite or extremely large (continuous state spaces)
- Game solving on full system: $O(|S| \times |A|)$ or worse
- For continuous systems: Must discretize entire state space

Scalability Advantage

Factor	Shielding	Traditional
Environment state space	Independent	Bottleneck
Spec complexity	Direct cost	Direct cost
Abstraction size	O(	Q_M
Scalability	Exponential in spec size	Exponential in state space size
Real-time deployment	Feasible (small automata)	Often intractable

Why Shielding Scales Better:

In a robot navigation task:

- Full state: position (x, y), velocity (vx, vy), orientation θ, sensor readings, etc. = Continuous/huge
- Abstraction for safety: "near wall" vs "far from wall" = 2-4 states
- Specification: "never crash" = 1-2 state automaton

Shielding synthesizes from 2-4 × 1-2 = small game, not from the full high-dimensional space.

Question 12: Atari Seaquest Analysis

(a) Apparent Contradiction and LTL Resolution

The Seeming Paradox:

- ϕ_1: Must surface before oxygen depletes → incentivizes surfacing
- ϕ_2: Must not surface if haven't collected divers → forbids surfacing

LTL Resolves This:

ϕ_1: $G(\text{oxygen} > 0 \vee \text{surface})$
"Globally: oxygen remains or you surface"

ϕ_2: $G((\text{divers\_collected} = 0 \wedge \text{depth} > 0) \rightarrow \neg\text{surface})$
"Globally: if no divers collected and underwater, don't surface"

No Contradiction: Together they mean:

- IF divers collected: must surface before oxygen depletes
- IF no divers collected: can stay deep (spec allows it)
- The specs partition the action space based on game state

(b) Information the Abstraction Must Track

The abstraction needs labels L including:

- `oxygen_level`: {low, medium, high} or {<threshold, ≥threshold}
- `divers_collected`: {zero, nonzero}
- `depth`: {surface, underwater}

Minimal abstraction:

- States: Q_M with transitions for {oxygen, divers, depth} combinations
- At least 2×2×2 = 8 states

(c) Why Shielding Didn't Improve Learning (Fig. 13)

Possible Explanations:

1. **Specifications were already "obvious":** The DQN agent quickly learned not to run out of oxygen and avoid premature surfacing through negative rewards; the shield provided little novel constraint.
2. **Action space is small:** Seaquest has limited actions; the agent explored adequately without restriction.
3. **Reward structure already incentivized safety:** The game's built-in penalties for surfacing prematurely/oxygen depletion aligned with specs, so shield added redundancy.
4. **Shield didn't enable exploration:** In preemptive shielding, the agent still had adequate freedom to explore optimal policies.

Key Insight: Shielding most improves learning when:

- Environment is complex (large action space)
- Safety constraints are non-obvious
- Reward signal is sparse or misleading
- Exploratory agents would naturally hit safety violations

Seaquest didn't meet these conditions strongly.

Question 13: Grid World Analysis (Fig. 11, Right)

Why |rank_t| = 3 Without Penalty Learns Best

Setting: 15×9 grid world with moving opponent; only 2 regions to visit.

What is rank_t?

- Agent provides ranking of 3 preferred actions: (a_1, a_2, a_3)
- Shield picks first action in ranking that's safe
- If all 3 are unsafe, shield picks any safe action

Learning Advantage with Ranking

|rank_t| = 3 Enables Multiple Policy Updates:

In one step where all 3 ranked actions are unsafe:

- Agent can update Q-values for: a_1, a_2, a_3, and the forced-safe action
- Creates 4 policy updates from 1 environment step
- Accelerates learning by ~4x in unsafe situations

Without Penalty (Approach 2 from Section 5.2):

The agent receives reward r_t+1 even for unsafe actions that were corrected:

- No punishment for trying unsafe moves
- Agent can assign reward to unsafe action: "this action was good here"
- Over time, learns that unsafe actions sometimes get corrected but lead to rewards
- Explores more freely; less biased against certain states

With Penalty (Approach 1):

- Unsafe action a_1 gets punishment $r' < 0$
- Agent learns to avoid a_1 at this state
- Less exploration, but more conservative policy
- Results in suboptimal final policy (Fig. 11 right, solid lines plateau lower)

|rank_t| = 1 Problem

With only 1 ranked action:

- If that action is unsafe, only 2 Q-updates (the unsafe one + forced safe)
- Less parallel learning
- Agent explores less efficiently

Safety Guarantee Maintained

Regardless of ranking size, the shield ensures:

- Safety is never violated
- Agent never executes unsafe action
- Learning can be "free-wheeling" within safety bounds

Question 14: Related Work Comparison

Teacher-Guided RL vs. Shielding

Aspect	Teacher-Guided RL	Shielding
Advice Source	Human teacher (reactive)	Formal synthesis (computed)
When Advice Given	When teacher decides OR learner fails	Precomputed for every state
Scalability	Requires human presence	Automatic, scales to large systems
Formality	Informal, heuristic	Formal guarantees (correctness proof)
Reward Manipulation	Teacher manipulates reward signal	Shield acts on action level
Convergence	Depends on teacher quality	Mathematically guaranteed
Applicability	Good for interactive learning	Better for autonomous systems

Advantages/Disadvantages

Teacher-Guided Strengths:

- Can incorporate human intuition/domain knowledge
- Can adapt to unexpected scenarios
- Natural for human-in-the-loop scenarios

Teacher-Guided Weaknesses:

- Doesn't scale (humans can't be online always)
- Informal; no correctness guarantee
- Manipulating rewards can mislead learning
- Expensive (labor-intensive)

Shielding Strengths:

- Fully automatic once abstraction/spec defined
- Formal correctness guarantees
- Scales to large state spaces
- Works post-deployment without modification

Shielding Weaknesses:

- Requires formal specification (hard to write)

- Abstraction must be conservative (might be too restrictive)
- Synthesis can be expensive computationally
- Less flexible than human teachers for unexpected scenarios

When to Use Each:

- **Teacher-Guided:** Early stage development, human-in-loop needed
- **Shielding:** Critical systems, autonomous operation required, formal guarantees essential

---

Question 15: Preemptive vs. Post-Posed Trade-offs

Scenario 1: Prefer Preemptive Shielding

Context: Industrial robot learning to assemble parts on a production line

Why Preemptive:

1. **Access to learning algorithm:** Manufacturing company has full system control
2. **Continuous feedback:** Robot benefits from knowing which actions are safe beforehand
3. **Faster learning:** Agent explores only safe region, no wasted unsafe actions/corrections
4. **Efficiency:** Assembly line has time pressure; faster convergence is critical
5. **Adaptation:** Agent can learn which safe actions are best; restricted but efficient

Setup:

- Abstraction: robot position zones (safe/collision zones)
- Spec: never enter collision zones
- Agent: Q-learning with restricted action set each step

Scenario 2: Prefer Post-Posed Shielding

Context: Deployed autonomous vehicle with pre-trained neural network policy

Why Post-Posed:

1. **Black-box agent:** Don't want to modify deployed system (third-party software)
2. **Minimal integration:** Shield is external layer; minimal system modification
3. **Retrofit safety:** Apply safety retrospectively without agent retraining
4. **Action ranking:** Vehicle already prioritizes maneuvers; shield respects priorities
5. **Robustness:** Even if agent fails, shield catches unsafe actions

Setup:

- Agent: pre-trained DNN that outputs steering commands ranked by confidence
- Shield: monitors commands in real-time
- Abstraction: discretizes road state (lane position, obstacle proximity)
- Spec: stay in lanes, maintain safe distance

Example Trace:

Step t:
Agent suggests: [slight-right, straight, left] (by confidence)
Shield checks: slight-right → unsafe (obstacle ahead)
Shield checks: straight → safe ✓
Shield executes: straight
Agent doesn't know it was corrected; updates policy with actual outcome

---

Comparison Summary

Dimension	Preemptive	Post-Posed
Integration	Tight (need learner modification)	Loose (external layer)
Learning Speed	Faster (no corrections)	Slower (actions get corrected)
Learning Quality	Better (learns from restricted space)	Potentially suboptimal
Deployment	New systems	Existing systems
Computational Load	Distributed to learner	Concentrated at shield
Flexibility	Lower (fixed safe action set)	Higher (respects rankings)
External Constraints	Can integrate spec requirements	Only enforces hard safety

---

Question 16: Fundamental Limitations Discussion

The Challenge Posed

The paper states: "To prevent unsafe actions, some approximate model of when actions are unsafe needs to be available."

Is This Fundamental?

YES, it is fundamentally necessary because:

1. **Information-Theoretic Argument:**
   - To distinguish safe vs. unsafe actions, shield must observe state
   - If state space is continuous/infinite, shield needs abstraction
   - Pure learning without model would require visiting unsafe states first
   - This is exactly what shielding prevents
2. **Impossible Alternative:**
   - Could we shield with zero prior knowledge? No.
   - Safe exploration requires knowing which actions *might* lead to danger
   - Without any model, can't predict consequences

Can It Be Overcome?

Partial Workarounds (but not elimination):

1. **Weaker Abstractions:**
   - Learn abstraction from agent experience

    - Start with permissive abstract model, refine it
    - Challenge: early learned abstractions might be wrong → unsafe
    - Risk: defeats purpose of guarantees
2. **Hybrid Approach:**
    - Start with loose/conservative abstraction
    - Gradually tighten as agent provides experience
    - Provide guarantees only for the learned portion
    - Limitation: can't guarantee safety in unexplored regions
3. **Assume Dynamics Provided:**
    - In many real systems, some dynamics are known (physics)
    - Use known model + learning for unknown parts
    - Limitation: only works if core safety dynamics are understood

The Unavoidable Core

The fundamental requirement: Some model of danger must exist before learning.

Why it can't be eliminated:

- Safety means "never do bad thing X"
- To know X is bad before experiencing it requires predictive model
- No pure learning system can guarantee safety without model

The paper's pragmatic stance:

This isn't actually a limitation—in safety-critical domains, constructing abstractions IS necessary and worthwhile because:

- Engineers already understand core safety concepts
- Writing formal specs is hard but feasible
- Cost of abstraction << cost of safety failure

Question 17: Shield State Space Calculation

Given Information

- Environment abstraction: 100 states (Q_M)
- LTL specification: 15-state automaton (Q)
- Plus error states

State Space Calculation

Product Construction:

- Game state space: G = Q × Q_M = 15 × 100 = **1,500 game states**

Then add:

- Unreachable states from initial (q_0, q_0,M): potentially reduced by reachability analysis
- Paradise states (where abstraction violated): 1 (usually merged)
- Error states: 1 (usually merged)

Approximate shield size: 1,500-2,000 states

Is This Problematic?

Not inherently problematic because:

1. **Shield operations are fast:** Finite-state machines are O(1) for each decision
2. **Synthesis time acceptable:** Game solving is O(|G| × |L| × |A|) ≈ O(1500 × 10 × 5) = manageable
3. **Memory footprint:** Small for modern computers
4. **Runtime execution:** Real-time capable

When it becomes problematic:

If spec has **50 states** and abstraction has **1000 states**:

- Game size: 50,000 states
- Still tractable, but pushing limits

If spec has **100 states** and abstraction has **10,000 states**:

- Game size: 1,000,000 states
- Approaching computational limits; synthesis might take seconds

Mitigation Strategies

1. **Refine Specification:**
   - Decompose complex LTL formula into simpler parts
   - Use modular synthesis (synthesize multiple small shields)
   - Example: Instead of one 50-state automaton, use two 10-state automata
2. **Coarsen Abstraction:**
   - Reduce abstraction granularity (merge similar states)
   - Only track features relevant to safety spec
   - Example: Instead of 1000 states, abstract to 50 relevant states
3. **Hierarchical Approach:**
   - High-level shield for critical safety (simple abstraction)
   - Low-level shields for detailed constraints
   - Compose them together
4. **Incremental Synthesis:**
   - Synthesize shields offline (not real-time)
   - Cache results
   - Update when spec changes (not frequently)

Question 18: Action Restriction Probability Calculation

Given Setup

- **Total actions:** 5
- **Safe actions (by shield):** 2
- **Unsafe actions (by shield):** 3
- **ε-greedy exploration:** ε = 0.1 (10% random, 90% exploit)

Without Shielding

Scenario: Agent learned Q-values; uses ε-greedy

- With probability 0.9: agent picks argmax(Q) → likely unsafe if poorly explored
- With probability 0.1: picks uniformly from 5 actions
  - Probability of unsafe in random: 3/5 = 0.6

Total probability of unsafe action:

If agent's greedy choice is unsafe (likely given poor exploration):

$$P(\text{unsafe}) = 0.9 \times (\text{probability greedy is unsafe}) + 0.1 \times 0.6$$
$$\approx 0.9 \times (\text{high}) + 0.06$$
$$\approx 0.5\text{-}0.9 \text{ (depending on exploration)}$$

Assuming half the learned actions are unsafe (due to incomplete exploration):

$$P(\text{unsafe}) \approx 0.9 \times 0.5 + 0.1 \times 0.6 = 0.45 + 0.06 = 0.51 \text{ (51\% unsafe)}$$

With Shielding (Preemptive)

Scenario: Shield restricts to 2 safe actions only

- Agent always chooses from $\{a_1, a_2\}$ (the safe 2)
- With probability 0.9: picks best of $\{a_1, a_2\}$
- With probability 0.1: picks uniformly from $\{a_1, a_2\}$

Total probability of unsafe action:

$$P(\text{unsafe}) = 0 \text{ (impossible—shield forbids unsafe actions)}$$

Comparison

Metric	Unshielded	Shielded
P(unsafe action)	~51%	0%
Reduction	Baseline	100%
Learning Space	5 actions	2 actions
Constraint	None	Minimal

Why This Calculation Matters

This shows the **safety benefit quantitatively:**

- **Unshielded risk:** Over 1000 steps, ~510 unsafe actions attempted
- **Shielded risk:** Over 1000 steps, 0 unsafe actions

For critical systems (robots, vehicles), this 51% → 0% reduction is massive.

Additional Consideration: Impact on Learning

Trade-off:

- Unshielded: Can explore all 5 actions, but 51% lead to unsafe outcomes
- Shielded: Constrained to 2 actions, but can explore those 2 optimally

In most cases, shielding **speeds up learning** because:

1. Agent wastes fewer steps on unsafe dead-ends
2. Focused exploration on safe region converges faster
3. Fewer penalties/resets from safety violations

This is precisely what Figures 11 and 14 demonstrate empirically.

Summary Table: All Questions at a Glance

Q #	Topic	Key Insight
7	Water Tank Design	Abstraction must track valve history; LTL captures temporal constraints
8	Game Synthesis	Winning region = states where system can force safety
9	Correctness	Inductive proof using W ensures spec satisfaction
10	Minimal Interference	Counterexample MDP proves action must be forbidden
11	Complexity	Shields scale with spec/abstraction, not full state space
12	Seaquest	LTL partitions action space; shielding didn't help due to redundancy
13	Grid World	Ranking + multiple updates = faster learning
14	vs. Teacher-Guided	Shielding: automatic/formal; Teacher-guided: flexible/human
15	Deployment Choice	Preemptive (new systems), Post-posed (existing systems)
16	Limitations	Abstraction fundamentally necessary, not eliminable
17	Shield Complexity	1,500 states acceptable; mitigate via spec/abstraction refinement
18	Safety Probability	51% unsafe → 0% unsafe; quantifies shielding benefit