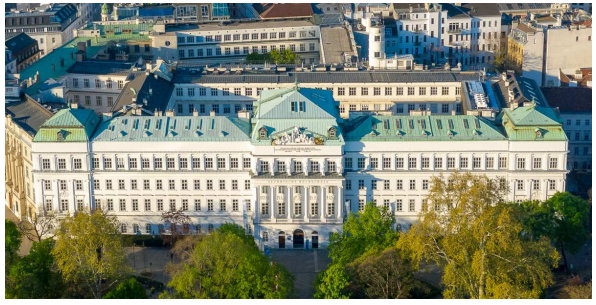
ASP for Reinforcement Learning

Thomas Eiter



Institute of Logic and Computation
Knowledge-Based Systems Group
Vienna University of Technology (TU Wien)

18th International Conference on Logic Programming
and Non-Monotonic Reasoning (LPNMR 2026)
September 5-11, 2026 Klagenfurt, Austria

TU WIEN kbs WWTF FWF Österreichischer Wissenschaftsfonds BILAI Bilateral Artificial Intelligence

Collaborators







Sebastian Adam, Rafael Bankosegger**, Ignazio Lopez, Martin Tappler, Ezio Bartocci, Johannes Oetsch*, Adrian Schmitt





Francesco Calimeri, Luca Laboccetta, Simona Perri, Giorgio Terracina

Advances of AI through Machine Learning

- Dramatic advances in AI, last decade(s) — driven substantially by Machine Learning
- Deep Learning at the center:



ChatGPT



DALL·E



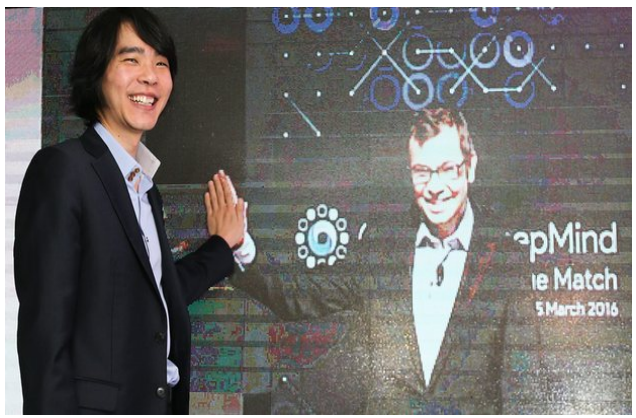
AlphaFold

All Deep Learning successes — none is Reinforcement Learning (RL)

RL: Learning from Data in Action Environments

- RL: learning *from interaction* — agent acts, environment responds (not a fixed labeled dataset)
- Large, diverse field — draws on operations research as much as ML; many variants (Sec. 2)
- Increasingly blended with deep learning — Deep RL
- Example: AlphaGo — Monte-Carlo Tree Search + deep neural networks

  [Silver *et al.*2016]





Potential for Improvement

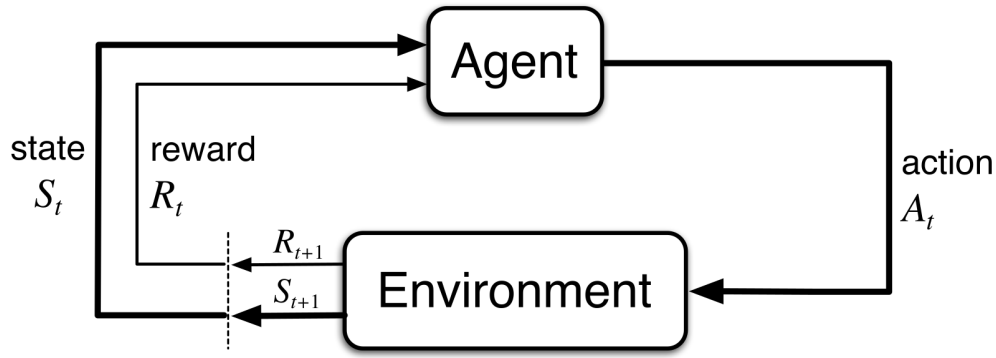
- Plain (model-free) RL: learns from reward alone
  — no explicit knowledge of world dynamics
- Model-guided RL: symbolic model with semantics provides
  - information about actions' effects
  - impossibility of actions
  - lookahead
- Bring Answer Set Programming into the picture:
  - expressive language
  - nonmonotonic negation, recursion
  - planning, reasoning, optimization

Outline of the Talk

1. Introduction
2. Reinforcement Learning
3. ASP and RL
4. Policy Management
5. Increasing Scalability
6. Abstraction
7. Conclusion

2. Reinforcement Learning

RL: Idea and History



- Agent–environment loop, discrete steps $t = 0, 1, 2, \ldots$: state S_t, action A_t, reward R_{t+1}, next state S_{t+1}
- Goal: policy π ($S_t \mapsto A_t$) maximizing cumulative (discounted, expected) reward — no supervision on "correct" actions, only (often delayed) reward → credit assignment problem
- **Bellman equation** [Bellman1957] — value of a state recursively via immediate reward + discounted value of successor:

$$V^\pi(S_t) = \mathbb{E}\left[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t, \pi\right]$$

- Standard assumptions ⇒ RL methods provably converge to the optimal policy
- Two flavors: model-free (learn directly from experience) vs. model-guided/-based (use a model of the dynamics for planning/lookahead → symbolic models, later)

Markov Decision Processes

- MDP: tuple (S, A, P, R, γ) — states S, actions A, transition probability $P(S_{t+1} \mid S_t, A_t)$, reward function R, discount $\gamma \in [0, 1)$
- Markov property: next state/reward depends only on current state/action, not full history

$$P(S_{t+1} \mid S_t, A_t) = P(S_{t+1} \mid S_t, A_t, S_{t-1}, A_{t-1}, \dots)$$

- Agent history: $S_0, A_0, R_1, S_1, A_1, R_2, S_2, \dots$
  - π may be history-dependent in general, but Markov property $\Rightarrow$ optimal π can be memoryless
- Common language for everything that follows: episodic/relational variants, Q-learning, later ASP-based models

Episodic and Relational MDPs

- Episodic MDPs: interaction split into episodes, each ending in a terminal state
  - history restarts on termination (goal / failure / time limit)
  - contrast: *continuing* tasks, no natural endpoint
  - most examples in this talk (Frozen Lake, Gardener) are episodic
- Relational MDPs: extend MDP with relational structure
  - actions parameterized, e.g. $move(x, y)$
  - state space: Cartesian product of attributes, finite or infinite domains
  - close to how ASP represents domains — objects, relations
- Relational structure $\rightarrow$ lets a symbolic/ASP model describe the environment compactly (bridge to Sec. 3)

Q-Learning

- Popular model-free, tabular method — action-value function $Q(s,a)$, no explicit P, R needed
- Target policy π_t (approaching optimal π^*) vs. behavior policy π_b (used to act/explore) — Q-learning is off-policy [Watkins and Dayan1992]
- Update rule (Bellman equation, action-value form):

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(S_{t+1},a) - Q(S_t,A_t)\right]$$

  α : learning rate (customary in machine learning for control)
- Exploration vs. exploitation: ϵ-greedy — random action with probability ϵ, else greedy
- Convergence guaranteed under standard conditions, but table $Q(s,a)$ needs storage for every state-action pair $\rightarrow$ scalability issue

Deep Q-Learning

- Problem: tabular Q-learning breaks down as $S \times A$ grows
  - explicit table infeasible even for large finite spaces
  - outright infeasible for continuous/infinite ranges
- Idea: function approximator — deep network $Q_\theta(s, a)$, weights θ
  - generalizes across similar states, rather than storing each
  - trained via gradient descent on the same Bellman error
- Standard method: DQN [Mnih *et al.*2015] — experience replay, target networks
- Trades exact convergence guarantees for scalability
- Deep RL setting: most ASP-guided approaches in this talk operate here — e.g. DQfD (deep Q-learning from demonstrations) reappears in Sec. 4

3. ASP and RL

Answer Set Programming — Recap

- ASP in a nutshell [Brewka *et al.*2011], [Lifschitz2019]: declarative LP, stable model / answer set semantics by Gelfond and Lifschitz [1988], [1991]
  - program describes a problem; answer sets are the solutions
  - nonmonotonic negation, recursion, disjunction — defaults, choice, combinatorial structure
- Language: first-order vocabulary; rules schematic, grounded for evaluation
- Generic rule form:

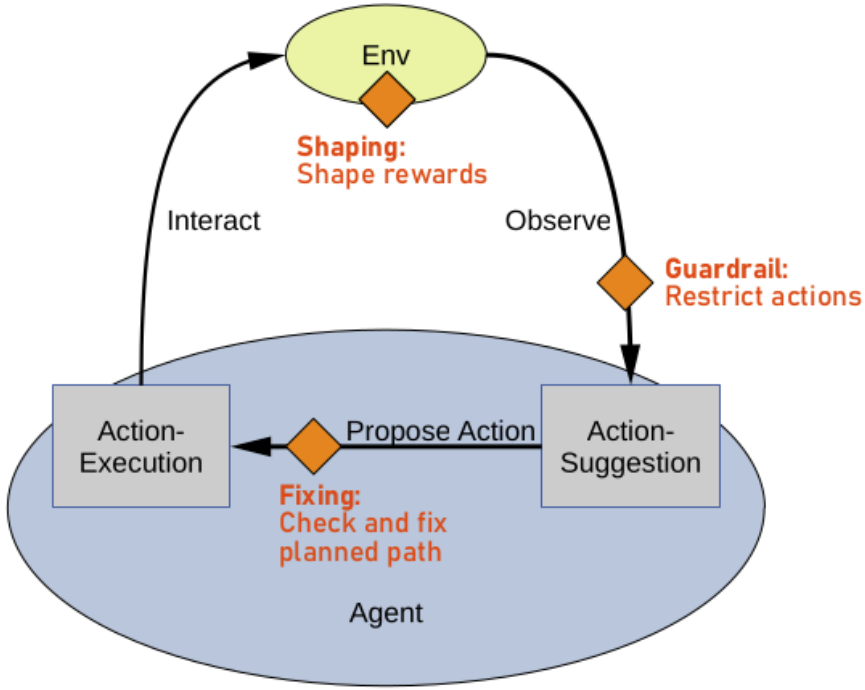
$$a_1 \vee \cdots \vee a_k \leftarrow b_1, \ldots, b_m, \mathbf{not}\ c_1, \ldots, \mathbf{not}\ c_n.$$

- Constructs: facts ($m{=}n{=}0$), rules, integrity constraints ($k{=}0$), choice rules, aggregates, weak constraints
- Systems: clingo [Gebser *et al.*2014], *dlv* [Leone *et al.*2006], *wasp* [Alviano *et al.*2013]
- Role of ASP in this talk: symbolic modeling and reasoning layer, guiding / constraining / checking RL agents

ASP as a Tool for RL

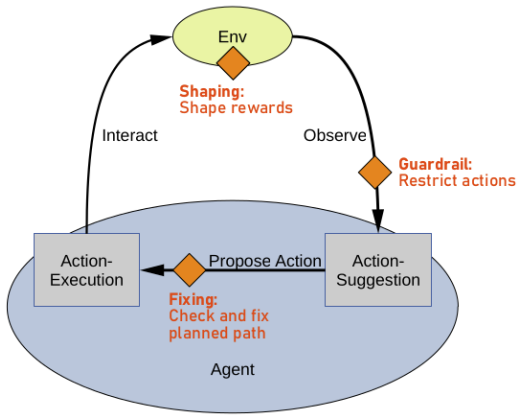
- **Opportunities**
  - predicate language — compact, structured states/actions
  - recursion, inertia — persistent facts, derived effects over time
  - optimization, aggregation, preference — weak constraints, priorities
  - incremental evaluation — multi-shot solving [Gebser *et al.*2019]
- **Challenges**
  - scalability — grounding blow-up, combinatorial search
  - large state/action spaces — where RL lives, where ASP traditionally struggles
  - interfacing neural networks — discrete/symbolic vs. continuous/sub-symbolic
  - numerics — ASP strong on integers, weak on continuous quantities
- Tension — expressive power vs. scalability — addressed in Secs. 4–6

ASP-for-RL Framework



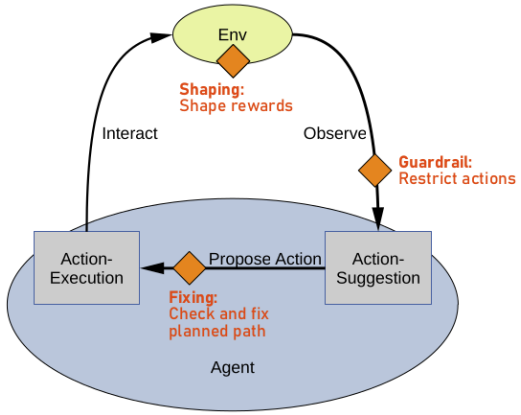
- Standard agent–environment loop, annotated with three runtime intervention points: [Schmitt2025], Fig.3.5
  - Shaping: shape rewards from the environment
  - Guardrail: restrict available actions, based on observation
  - Fixing: check/fix the planned path before execution
- Fourth possibility, outside runtime dynamics:
  - Abstraction: transforms the underlying model (state/action space), not a point in the cycle
- Together: a useful taxonomy for structuring prior and own work

Brief History of ASP-for-RL



- Nickles [2011] [2012] — RRL: RL + ASP; several grounders/solvers, no planning *(deprecated)* → shaping
- Leonetti et al. [2016] — DARLING: standard ASP constrains RL exploration; deterministic domain model → guardrail
- dos Santos et al. [2022] — relaxed spatial puzzles; ASP (domain) + MDP → heuristics for policy construction → guardrail-style

KBS@TU Wien — ASP for RL Line of Work



- Datler [Datler2020]: ASP domain model + planning, guides exploration vs. MCTS *(blocks world)* → guardrail
- Hatschka [Hatschka *et al.*2023]: norms in ASP, normative supervisor *(ethical Pacman)* → guardrail
- Schmitt [Schmitt2025]: nondeterministic FrozenLake, ASP planning (telingo) + RL → fixing, guardrail, shaping
- Adam [Adam and Eiter2025], [Lopez-Miguel *et al.*2025]: policy fixing under post-training norms → fixing, shaping
- Bankosegger [Bankosegger *et al.*2026]: ASP for abstraction in RL → abstraction
- This talk: focus on policy fixing, abstraction

Normative ASP

- Context: deontic logic / deontic ASP — obligation, permission, prohibition
- Key point: a norm is a soft constraint — violable, at a price
- Example contrary-to-duty (CTD) conflict [Prakken and Sergot1996]

$$F(kill) \qquad O(kill_softly \mid kill)$$

second norm activates only once the first is violated

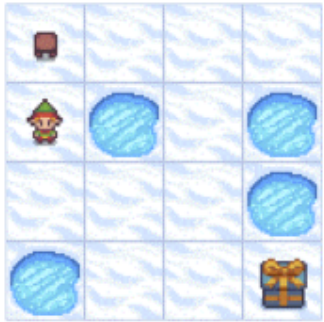
- Representing norms in ASP:
  - direct ASP encoding: Hatschka et al. [2023] (weak constraints); Governatori [2024] (defeasible deontic logic); Shams et al. [2017] (application-specific)
  - deontic ASP extensions: Gonçalves & Alferes [2013] (deontic logic programs); Cabalar et al. [2023] (DELX)
  - deontic temporal ASP: Giordano et al. [2013]; Soldà et al. [2025]

Weak-constraint encoding of the CTD example [Hatschka *et al.*2023]:

$$:\sim \ Do(kill). \qquad [1:2]$$
$$:\sim \ Do(kill), notDo(kill_s oftly). \qquad [1:1]$$

- higher level (2) prioritizes not killing;
- lower level (1) still requires kill-softly *if* a kill occurred — no inconsistency

Adrian Schmitt's Work — Deontic FrozenLake

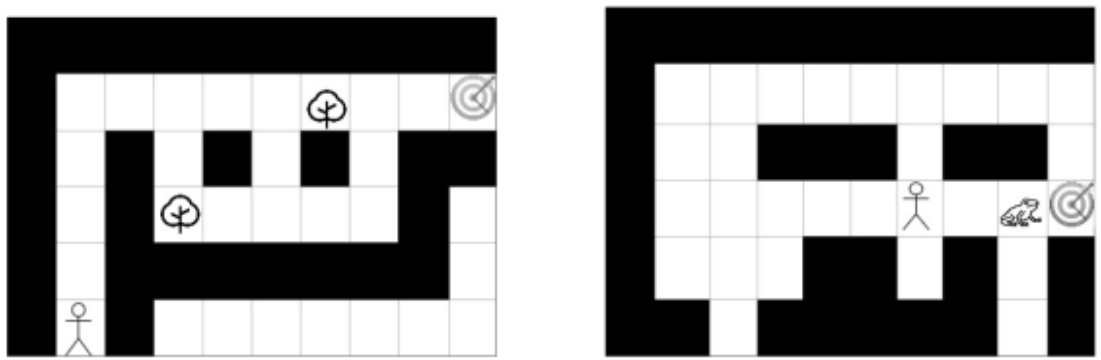


S	F	F	F
F	H	F	H
F	F	F	H
H	F	F	G

- Environment [Schmitt2025]: FrozenLake (4×4), extended with moving *traverser* and *presents*, under deontic norms
- Scope: five experiment suites for learning/planning parameters, norm-sets/CTDs, deontic conflicts, and enforcement strategies
- Key findings:
  - static vs. dynamic/complex norms: progress & static norms learned reliably; CTDs & dynamic norms harder
  - planning-based enforcement: speeds up learning, norm-adherent paths — but blind to dynamic aspects
  - reward-shaping: learns all norm types (risk: local optima w/o planning help)
  - norm-based Q-table initialization: also performs well
- Takeaway: no single strategy dominates — combining planning-based reasoning with reward-shaping looks most promising

4. Policy Management

Policy Fixing — The Gardener Example



(a) Deterministic (b) Nondeterministic

- Gardener: agent moves toward target on a grid (N/E/S/W); cells may hold plants or frogs — moving onto one kills it
  - plants stationary $\rightarrow$ deterministic (a); frogs move arbitrarily $\rightarrow$ nondeterministic (b) — must account for *different possible evolutions*
- Norm: obligation to never kill a living entity — $O(do_not_kill)$
- Problem: plain-RL policy π (reward = reach target fast) is norm-*agnostic*
- Policy fixing: locally alter π's suggested actions to avoid violation, staying as close as possible to π's preferences — rather than costly retraining

Policy Fixing — Formal Notions

- Setup: π violates maintenance norm Φ at t; violation in s at t' costs penalty $p(\Phi, s, t') \geq 0$
- Policy fix (strict): π', no worse than π by Q-values ($\pi' \preceq_Q \pi$), minimizing worst-case future penalty:

$$\pi' = \operatorname{argmin}^{\preceq_Q}_{\pi'' \in \Pi} \max_{\sigma \in Tr(\pi'', t)} \sum_{t' > t} p(\Phi, s_{t'}, t')$$

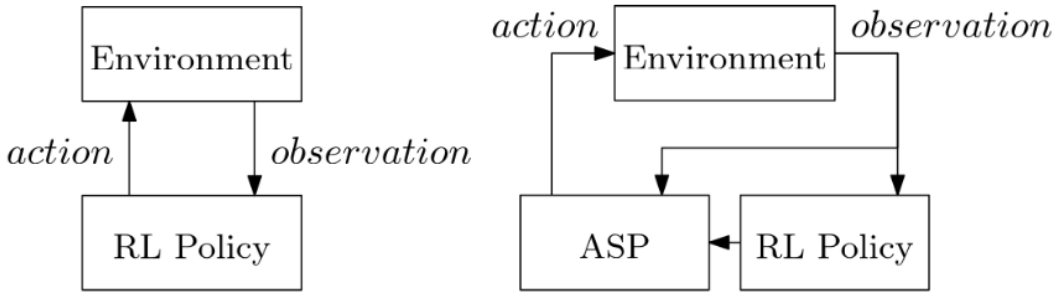
- Utility-based fix: jointly optimize reward and norms — replace outer argmin by argmax over expected reward minus penalty:

$$\pi' = \operatorname{argmax}_{\pi'' \in \Pi} \; r(\pi'', s_t, t) \; - \; \max_{\sigma \in Tr(\pi'', t)} \sum_{t' > t} p(\Phi, s_{t'}, t')$$

  norm violation now a discount on reward, not an absolute priority

- k-policy fix: full fixing is a planning problem, but horizon growth $\rightarrow$ exponential blow-up — approximate: fix only next k steps, recompute periodically

Policy Fixing Framework



(a) Traditional RL framework (b) ASP enhanced framework

- Observation feeds both RL policy and ASP component; RL passes action preferences to ASP, which selects the executed action — a k-policy fix, a conformant plan of length k
- Conformant planning: goal reached for *every* possible evolution — EXPSPACE-complete in general [Rintanen2004], Σ_2^P-complete for polynomial k [Baral *et al.*2000]
- Guess and check:
  - P_{gen}: candidate trajectories (weak constraints: norm penalty + RL preference);
  - P_{check}: *no* evolution violates the norm — use saturation technique [Eiter *et al.*2009]

Policy Fixing — ASP Encoding (P_{gen})

Listing 1: Excerpt of P_{gen}

```
1  plant(4,4).
2  player(0,2,6).
3  p_n(T)|p_s(T)|p_w(T)|p_e(T) :- T=1..k.
4  player(T,C',R) :- player(T-1,C,R),
⋮      p_w(T), C'=C-1.
8  :~ player(T,C,R), plant(C,R). [1@2]
9  #maximize {R@1,T : reward(R,A,T)}.
```

- Static facts (plant/player positions)
- Disjunctive action rule: one action guessed per time step ($T \leq k$)
- Transition rules for the guessed actions
- Weak constraint penalizing landing on a plant (line 8, level 2)
- Optimization statement following the RL policy's reward preference, lower priority (line 9, level 1)

Policy Fixing — ASP Encoding (P_{check})

Listing 2: Excerpt of P_{check}

```
10  frog(0,6,2).
11  f_n(T)|f_s(T)|f_w(T)|f_e(T) :- T=1..k.
12  frog(T,C',R) :- frog(T-1,C,R), f_w(T),
:       C'=C-1.
16  ok(0).
17  ok(T) :- player(T,C,R), frog(T,C',R'),
:       C'!=C, ok(T-1).
19  :- not sat.
20  sat :- ok(k).
21  f_n(T) :- T = 1..k, sat.
:
25  sat :- frog(T,C,R), wall(C,R).
26  good(T) | bad(T) :- T=1..k.
27  ok(T) :- bad(T).
28  #minimize {1@3,T : bad(T)}.
```

- Mirrors P_{gen}'s structure for the environment's nondet. elements (frog)
- Guesses possible frog moves; derives compliance via helper atom `ok(T)`
- Saturation technique (19-21), (25-28) enforces "*no* guessed evolution violates the norm" telingo not applicable!
- Penalties folded in via a `good/bad` guess-and-minimize template — naive weak constraints alone would be falsified by saturation
- Priority: norm penalties (level 2) always outrank reward preference (level 1)

Optimizations

- Multi-shot solving [Gebser *et al.*2019]:

  `#external` atoms, set dynamically via clingo's API — avoids re-grounding at each step; ground all possible initial states once, up front

- Dimensionality reduction:

  split multi-argument atoms (e.g. $player(T, C, R)$) into independent ones — cuts grounded rules for comparisons

  $ok(T) \leftarrow player(T, C, R), frog(T, C', R'), C \neq C' \quad \Rightarrow \quad (c \cdot r)^2 \rightarrow c^2$ rules

- Windowing:

  encode only state within radius r of the agent — roughly constant number of grounded rules as instance size grows, at the cost of ignoring outside-window effects

Experiments — Gardener

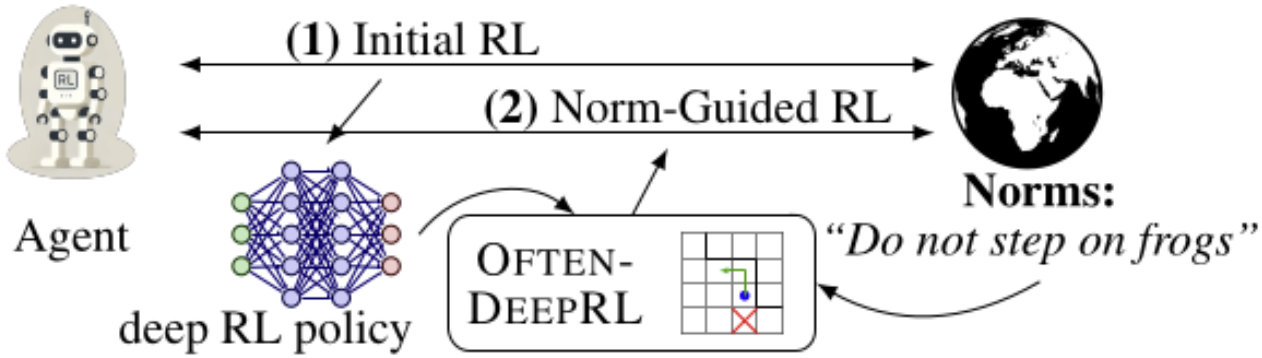
Config	1-Step kills plants / frogs	1-Step avg. / startup time in ms	2-Step kills plants / frogs	2-Step avg. / startup time in ms
2500 walls, 1000 plants and 0 frogs				
RL	11.2 / 0	0 / 0	11.2 / 0	0 / 0
r=3, k=1	6.0 / 0	179 / 180	NA	NA
r=3, k=2	5.4 / 0	182 / 196	4.8 / 0	91 / 187
r=3, k=4	3.5 / 0	198 / 256	3.3 / 0	98 / 248
r=3, k=6	3.5 / 0	283 / 414	3.3 / 0	140 / 413
r=5, k=1	6.0 / 0	268 / 258	NA	NA
r=5, k=2	5.3 / 0	270 / 266	4.8 / 0	134 / 264
r=5, k=4	3.5 / 0	283 / 333	3.6 / 0	143 / 336
r=5, k=6	2.7 / 0	411 / 572	3.1 / 0	203 / 569

Config	1-Step kills plants / frogs	1-Step avg. / startup time in ms	2-Step kills plants / frogs	2-Step avg. / startup time in ms
2500 walls, 250 plants and 250 frogs				
RL	4.00 / 2.80	47 / 46	4.00 / 2.80	47 / 46
r=3, k=1	2.02 / 1.24	284 / 359	NA	NA
r=3, k=2	1.74 / 1.14	297 / 505	1.56 / 1.24	173 / 507
r=3, k=4	0.96 / 0.78	469 / 1360	0.92 / 0.80	257 / 1367
r=3, k=6	0.86 / 0.64	1438 / 3545	0.64 / 0.82	727 / 3826
r=5, k=1	1.90 / 1.12	424 / 954	NA	NA
r=5, k=2	1.80 / 0.82	508 / 2312	1.70 / 1.08	280 / 2381
r=5, k=4	0.90 / 0.76	1424 / 12592	0.84 / 0.74	720 / 12983
r=5, k=6	1.02 / 0.66	6528 / 54842	0.98 / 0.74	3334 / 55767

- Setup: 80 instances (det./nondet.), varying grid size, plant/frog density; vs. norm-agnostic RL baseline
- Even $k{=}1$ drastically reduces violations; increasing r, k together improves further (best: $r{=}5, k{=}6$)
- Nondet. case markedly more expensive — coNP check over frog evolutions
- First 2 steps of a k-fix: roughly halves runtime, little compliance loss
- Also validated: Pacman (vegan/vegetarian norms), CCN caching — pattern generalizes beyond Gardener

Policy fixing overrules the learned policy at each violation — can the norm knowledge be folded back into the policy itself? (next)

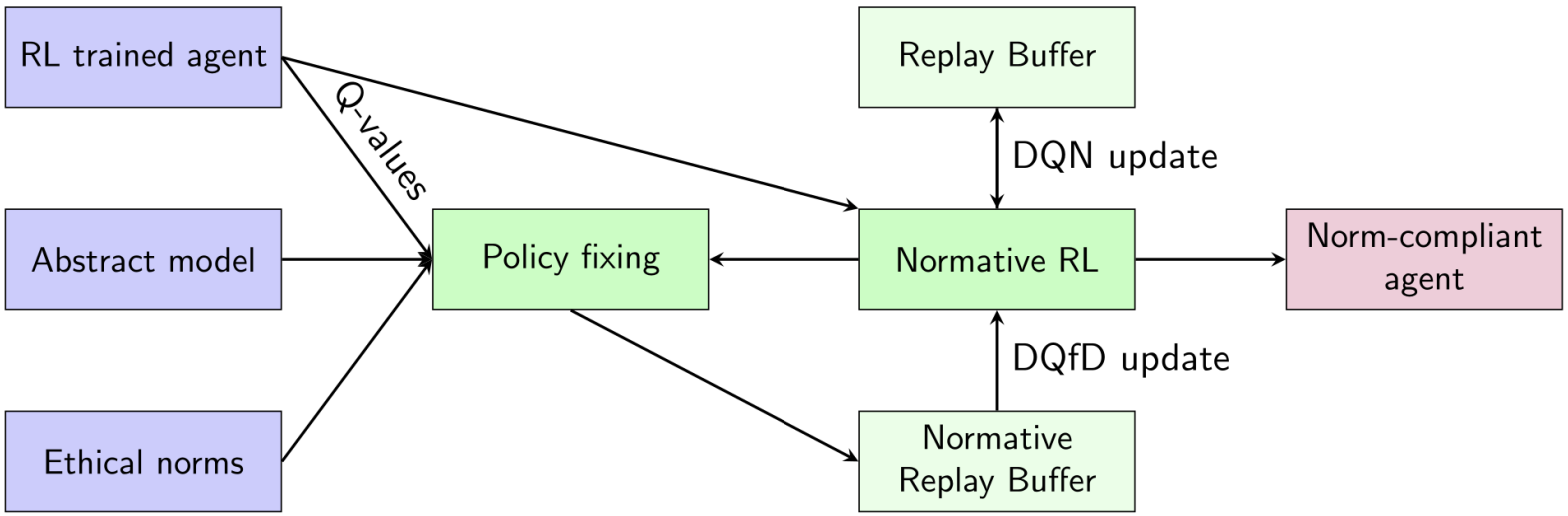
Training on the Fly (OFTEN-DeepRL)



[Lopez-Miguel *et al.*2025]

- Motivation: runtime policy fixing costs planning time repeatedly *and* never changes what the neural policy has learned
- Idea: use policy fixes as training data — internalize the norm via demonstration learning (DQfD [Hester *et al.*2018])
- Pipeline: (1) Initial RL — train DQN on task alone; (2) Norm-Guided RL — ASP-computed policy fixes become demonstrations, fine-tune the policy
- Neurosymbolic loop: neural $\rightarrow$ symbolic (ASP model) $\rightarrow$ neural (final norm-compliant policy)

Deep RL Learning — Norm-Guided Training Pipeline



- Two experience streams, two replay buffers: standard $\rightarrow$ DQN update; normative (policy-fix actions, as *expert demonstrations*) $\rightarrow$ DQfD update
- DQfD: expert/hinge loss J_E pushes the demonstrated action's Q-value above others by a margin
- Combined loss:

$$J(Q,N,S) = J_{DQN}(Q,S) + J_{DQN}(Q,N) + \lambda_n J_E(Q,N)$$

λ_n increases gradually: reward-max. $\rightarrow$ norm-compliant

Experiments — Gardener

Experiment size-f, p	Base			OFTEN-DEEPRL			Policy Fixes		
	SR	rel-steps	frogs / plants	SR	rel-steps	frogs / plants	SR	rel-steps	frogs / plants
25×25 - $p = 0.05, f = 0.5$	1.00	1.09	0.53 / 2.45	1.00	1.07	0.30 / 0.87	0.15	1.43	0.66 / 0.00
25×25 - $p = 0.05, f = 0$	0.99	1.49	— / 4.40	1.00	1.13	— / 0.53	0.00	—	— / 0.50
25×25 - $p = 0.05, f = 1.0$	1.00	1.10	1.26 / —	0.99	1.11	0.79 / —	0.46	2.44	1.39 / —
50×50 - $p = 0.05, f = 0.5$	1.00	1.28	1.53 / 3.71	1.00	1.23	1.15 / 1.06	0.01	5.21	1.90 / 0.50
50×50 - $p = 0.05, f = 0$	1.00	1.43	— / 9.39	0.99	1.32	— / 1.73	0.00	—	— / 0.00
50×50 - $p = 0.05, f = 1.0$	1.00	1.12	2.86 / —	1.00	1.11	2.30 / —	0.14	3.20	3.08 / —

- SR = success rate; rel-steps = ratio to optimal; frogs/plants = kills/episode
- Base: SR ≈1.00, but kills regularly — norm-agnostic
- Policy Fixes: kills drop, but SR collapses (e.g. 0.15, 0.00, 0.46) — overridden too aggressively, agent gets trapped
- OFTEN-DeepRL: keeps Base-level SR *and* substantially reduces kills — internalized norm doesn't fight the task
- Same pattern on Pacman, SUMO traffic control — generalizes beyond Gardener

5. Increasing Scalability

ASP Streaming

- Recall: policy fixing repeatedly invokes a batch ASP solver
  - each step, current observation re-encoded as a fresh ASP problem, solved from scratch
- Scalability issue:
  - growing grid size, radius r, horizon k: re-grounding/solving bottleneck
  - an external procedural engine (Python) orchestrates calls, manually carrying time info across them
- Structural issue: this stateless interaction doesn't fit temporal norms
  - can't deal naturally with compliance depending on accumulated history (forget everything between calls)
- Alternative: ASP-based Stream Reasoning (SR) [Adam *et al.*2026]
  - treat observations as a genuine data stream, reasoned over incrementally and statefully
  - conclusions are continuously updated *and* retractable (nonmonotonicity essential)
  - use windows (last w time points) instead of full history

DP-sr Solution

- DP-sr [Laboccetta *et al.*2024] ASP-based SR framework
  - streaming literals: `L in [w]` (window of last w) vs. `L in {w}` (exactly w back)
  - inertial events (`#generate`): persist until deactivated (e.g. "plant stays killed")
  - external atoms (`&...`): interface with the RL policy's action preferences
  - aggregates over intervals: distinguishing feature among ASP-based SR engines; natural fit for *cumulative* norms
- Hybrid time management
  - past/current handled implicitly by the stream
  - future planning steps represented explicitly, relative `@now`
- Oil-Fire norm: stepping on oil $\rightarrow$ "inflammable" for K steps; entering fire while inflammable = severe violation

```
     % Activation: stepping on oil lasts for k steps
r26: inflammable(T) :- T=@now, p(R,C,T), oil(C,R).
r27: inflammable(T) :- inflammable(T) in {1}, T1=@now,steps_inflammable(K), T1-T <= K.
r28: activate_planning :- T=@now, p(R,C,T), oil(C,R).

     % Safety Constraint: avoid fire while inflammable
r29: penalty_fire(T) :- inflammable(T), p(R,C,T), fire(C,R).
r30: :~penalty_fire(T). [10@1,T]
```

`inflammable(T)` activates/persists via `in{1}`

- Similar Mud-Water norm also triggers a dynamic policy switch

Experimental Results

Grid Size	Config.	System	Steps	Time/Step (ms)	Plants Killed	Frogs Killed
50 × 50	R=3, H=4	Baseline	81.6	216.7	0.3	0.4
		DP-sr	66.8	742.9	0.1	0.1
	R=5, H=4	Baseline	84.0	505.7	0.2	0.9
		DP-sr	58.0	750.5	0.0	0.2
	R=5, H=6	Baseline	98.2	2649.0	0.4	0.7
		DP-sr	66.0	771.0	0.0	0.1
100 × 100	R=3, H=4	Baseline	111.8	447.9	1.9	1.4
		DP-sr	115.0	1069.9	1.5	1.6
	R=5, H=4	Baseline	112.8	1319.5	1.5	1.6
		DP-sr	120.0	1155.0	1.6	1.0
	R=5, H=6	Baseline	142.8	8554.5	2.4	2.4
		DP-sr	175.0	1176.3	1.7	0.9
150 × 150	R=3, H=4	Baseline	141.8	1113.6	1.7	0.9
		DP-sr	162.8	2013.8	1.7	1.6
	R=5, H=4	Baseline	148.0	2342.3	1.3	1.0
		DP-sr	178.0	2100.9	1.9	1.9
	R=5, H=6	Baseline	168.2	10663.3	2.3	2.0
		DP-sr	191.2	2151.3	1.9	1.9
	R=5, H=8	Baseline	200.6	49299.6	2.9	1.5
		DP-sr	186.8	2211.3	1.6	1.6
200 × 200	R=3, H=4	Baseline	224.2	2424.1	2.3	3.2
		DP-sr	206.6	4014.1	1.8	3.7
	R=5, H=6	Baseline	270.4	12735.8	2.9	5.1
		DP-sr	284.6	4291.0	2.4	3.0
	R=5, H=8	Baseline	312.4	56318.1	2.6	4.7
		DP-sr	262.0	4298.4	2.0	2.3

- 10-run averages, Baseline (batch) vs. DP-sr, grid sizes 50×50 to 200×200
- Norm compliance, mission efficiency: roughly comparable (no clear winner)
- System performance: for small grids Baseline competitive; as size and (r, k) grow, DP-sr scales dramatically better — e.g. 150×150, $r{=}5$, $k{=}8$: 49.3s $\rightarrow$ 2.2s/step ($\sim$96% reduction)
- Bottom line: DP-sr substantially mitigates (not eliminates) the horizon-driven scalability problem

Experimental Results

Grid Size	Config.	Steps	Time/Step (ms)	Mud Pen.	Fire Pen.	Plants/Frogs Killed
50 × 50	R=3, H=4	113.4	814.5	0.8	0.2	0.2 / 0.6
	R=5, H=4	111.0	829.6	1.1	0.0	0.2 / 0.3
	R=5, H=6	103.6	875.4	1.3	0.0	0.1 / 0.1
100 × 100	R=3, H=4	182.0	1349.7	0.0	2.1	1.5 / 1.0
	R=5, H=4	238.2	1421.0	0.0	1.8	2.0 / 1.6
	R=5, H=6	204.1	1429.1	0.0	2.8	1.4 / 1.0
150 × 150	R=3, H=4	190.2	2470.9	0.0	1.7	2.3 / 1.7
	R=5, H=4	193.2	2630.1	0.1	2.0	2.1 / 2.3
	R=5, H=6	217.6	2671.9	0.0	2.4	2.3 / 1.7

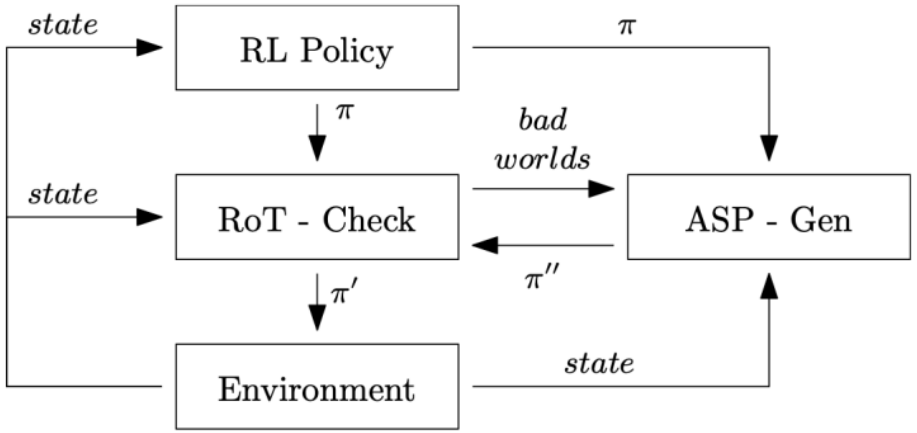
- Again, 10-run averages
- With temporal norms activated (Mud-Water, Oil-Fire) — only feasible with the streaming/stateful approach
- Time/step increases only moderately over Table 1's DP-sr numbers, mainly due to more frequent re-planning
- Mud/Fire penalties stay low across grid sizes; step counts/kills close to the non-temporal setting
- Framework absorbs added normative complexity without a proportional performance hit

Approximation

- Scalability challenge — why exact approaches struggle:
  - shielding [Könighofer *et al.*2025]:
    hard constraints (safe/unsafe) — but deontic norms are inherently *soft*
  - worst-case planning [Adam and Eiter2025]:
    safe for every evolution — but EXPSPACE-/Σ_2^pcomplete in general
  - probabilistic logic programming (exact probabilities via model counting):
    $\# \cdot$ coNP-complete for disjunctive programs — unrealistic at scale
- Idea — probabilistic policy fixing [Adam and Eiter2026]:

  relax “safe in every world” to “safe with high confidence, estimated by sampling”
  - a monitoring program evaluates the trajectory against a set W of sampled world evolutions
  - trigger emergency planning only if safety guarantees fail on this sample
  - fix computed via ASP on W — minimizes violations across sampled worlds, stays close to the reward-maximizing policy
- Proviso: still assumes an exact model of environment dynamics — sampling is over evolutions, not over model uncertainty itself

Statistical Model Checking Intervention

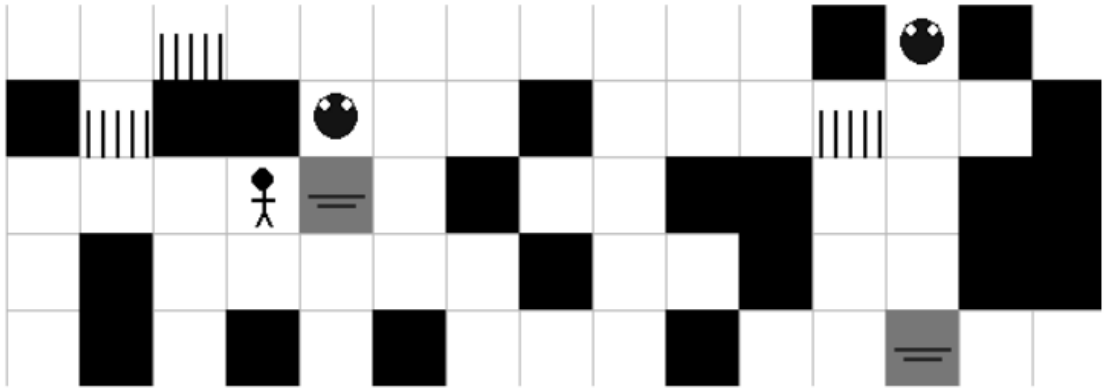


- Rule of Three (RoT) [Jovanovic and Levy1997]: if n sampled worlds show zero violations, the true violation probability is $\leq 3/n$ with 95% confidence
- Hoeffding's inequality [1963] (when some samples *do* violate):

$$P(|\hat{p} - p| \geq \epsilon) \leq 2e^{-2N\epsilon^2}$$

- Guarantee: with $N \geq (\ln 2 + \ln |\Pi| - \ln \delta)/2\epsilon^2$ samples, confidence $1-\delta$ that the computed policy is within 2ϵ of optimal
- Loop:
  - RoT-Check (normal ASP) tests a candidate against n fresh samples, returns "bad worlds" B;
  - ASP-Gen (disj. ASP + weak constraints) builds a new candidate – high priority: minimizing violations on B, low priority: Q-preference
  - repeat until success or budget N exhausted

Gardener (Puddle Variant) — Domain and Norms



- Domain [Adam and Eiter2026]: no fixed target — score by mowing grass (10pt), draining puddles (5pt); frogs bias 70% toward full puddles; draining stuns nearby frogs
- Environment reacts to the agent's own actions
- Norms:
  - MTN1: do not capture a frog
  - MTN2: do not drain a puddle with a frog nearby
  - CTD: if MTN2 is violated, capture that frog within 4 steps

Experiments — Gardener (Puddle Variant)

Method	Steps	Dev. π_Q	Time (ms)	MTN1	CTD [MTN2]
RL	165.90	—	—	0.80	0.32 [0.80]
UPF	195.70	21.80	3108.12	0.00	1.12 [1.40]
PPF 65-25	169.70	5.70	190.63	0.07	0.06 [0.90]
PPF* 65-25	169.00	8.00	198.36	0.00	0.00 [0.60]
PPF 90-10	167.40	4.20	295.75	0.00	0.00 [0.40]
PPF* 90-10	168.40	9.40	266.81	0.05	0.03 [0.80]
PPF 95-5	167.70	9.30	574.19	0.00	0.00 [0.70]
PPF* 95-5	175.90	15.50	489.87	0.00	0.00 [0.40]
PPF 99-5	168.40	9.10	713.78	0.00	0.00 [0.80]
PPF* 99-5	168.60	8.50	447.59	0.00	0.00 [0.70]
PPF 99-3	170.60	9.50	1201.53	0.00	0.00 [0.70]
PPF* 99-3	170.80	11.90	777.98	0.00	0.00 [1.00]

Table 3: Avg. 100 rounds, 100×100 grid, h=5, $(1-\delta)$–%ϵ conf.–err.

- Vanilla RL violates constantly; UPF [Adam and Eiter2025] drives MTN1 $\rightarrow$ 0 but at 3.1s/step, and *worsens* CTD (doesn't reason about the compound norm)
- PPF and PPF* (cache extension for policy fixes) achieve near-full compliance on all three norms at once, at a fraction of UPF's runtime
- Compliance degrades gracefully with relaxed confidence/error tolerance

6. Abstraction

Why Abstraction

- Value-based RL (e.g. Q-learning): learn $q : S \times A \mapsto \mathbb{R}$, apply greedy policy $\pi_q(s) \doteq \arg\max_a \{q(s,a)\}$
- Problem: combinatorial explosion in $|S \times A|$
  - e.g., $\Omega(n!)$ states in an n-blocks world alone
  - tabular q infeasible; even Deep RL needs large data, can be unstable
- Needed: compact, generalizable representations of q and π
  - group state-action pairs that "behave the same"
- State-Action Pair Abstraction: aggregate similar pairs into an abstract pair $(\hat{s}, \hat{a})$, with one estimated value $\hat{q}(\hat{s}, \hat{a})$ for all "block" members
- CARCASS (*Compact Abstraction using Relational Conjunctions for Aggregation of State-action Spaces*) [van Otterlo2009]

  build such abstractions via first-order logic:
  - ordered decision list: $State \rightarrow \{Action_1, \ldots, Action_n\}$
  - e.g. $clear(X), clear(Y), on(Y, 0) \rightarrow \{move(X, Y), move(Y, X)\}$
  - originally Prolog (SLDNF-resolution), redo in ASP [Bankosegger *et al.*2026]

CARCASS Abstraction

- Foundation [van Otterlo2009]: finite, ordered list $\hat{C} = \langle (\hat{s}_1; \hat{A}_{\hat{s}_1}), \ldots, (\hat{s}_n; \hat{A}_{\hat{s}_n}) \rangle$, $n \geq 1$
  - abstract state $\hat{s} \doteq l_1, \ldots, l_k$: conjunction of naf-literals
  - abstract action $\hat{a} \in \hat{A}_{\hat{s}}$: single atom $\hat{a} \doteq p(\overline{t})$
- Covering relation: $((s,a),(\hat{s},\hat{a})) \in Cov$ which abstract state/action a *concrete* pair falls under (decided by SLDNF-resolution)
- Decision-list semantics: to classify concrete s, walk the list in order, choose the first (smallest-index) $\hat{s}_i$ that covers s (Prolog-style)
- Example: Blocks World

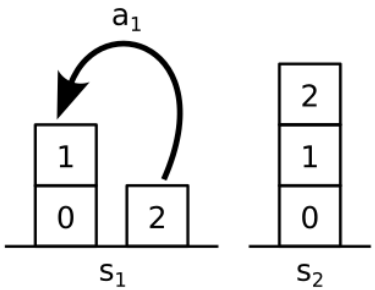
$$\big(clear(X), clear(Y), on(Y,0);\ \{move(X,Y), move(Y,X)\}\big)$$

covers every state where some X, Y are clear and Y sits on block 0

Encoding Abstractions in ASP

- Contribution [Bankosegger *et al.*2026]: given $\hat{C}$, define ASP encoding $P_{\hat{C}}$ that provably correct encodes Cov and the decision list
  - replaces Prolog/SLDNF with fully declarative ASP
  - enables to easily add ASP background knowledge P_B (with weak constraints allowed)
- Guess-and-check:
  - $sCov(\lambda(\hat{s}_i), \bar{V})$: $\hat{s}_i$ covers a concrete state
  - $sChoice(\lambda(\hat{s}_i))$: choice rule picks one covering state — the "guess"
  - $aCov(\lambda(\hat{s}_i, \hat{a}_{i,j}), p(\bar{t}))$: $\hat{a}_{i,j}$ covers a concrete action
  - a constraint eliminates all but the smallest index — the "check"
- Online interaction:
  - map state s, actions A_s to optimal answer set $I \in \mathcal{AS}(P_{\hat{C}} \cup P_B \cup s \cup A_s)$
  - extract $(\hat{s}, \hat{A}_{\hat{s}})$ and relevant Cov from it
  - feed to abstract Q-learning on $(\hat{s}, \hat{A}_{\hat{s}})$

Blocks World



$$s_1 = \left\{ \begin{array}{l} \mathbf{on}(0, table), \mathbf{on}(1, 0), \mathbf{on}(2, table), \\ \mathbf{goal}(0, table), \mathbf{goal}(1, 0), \mathbf{goal}(2, 1) \end{array} \right\}$$

$$a_1 = \mathbf{move}(2, 1)$$

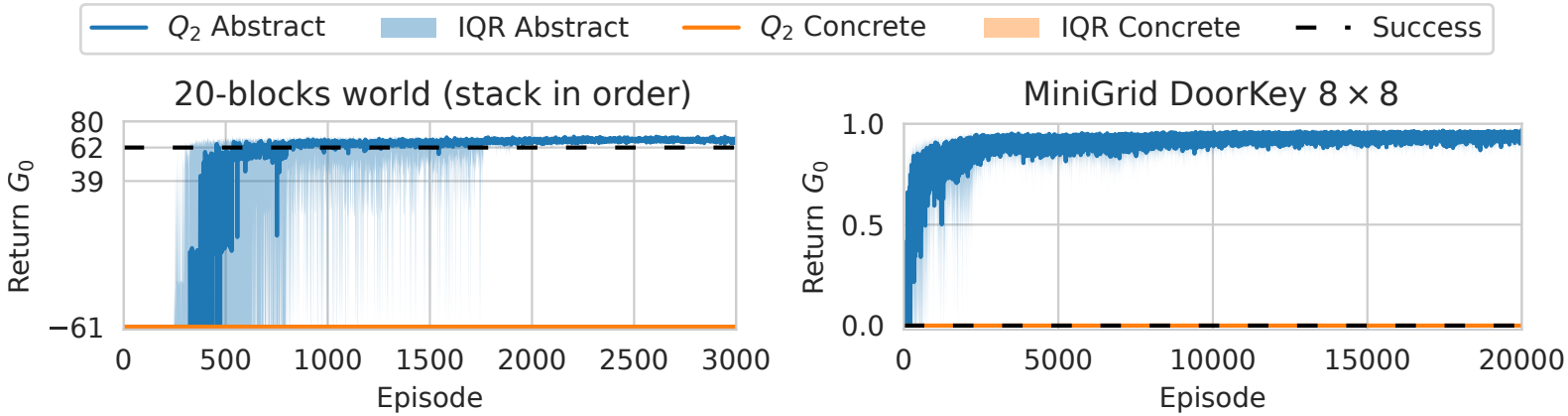
$$s_2 = \left\{ \begin{array}{l} \mathbf{on}(0, table), \mathbf{on}(1, 0), \mathbf{on}(2, 1), \\ \mathbf{goal}(0, table), \mathbf{goal}(1, 0), \mathbf{goal}(2, 1) \end{array} \right\}$$

- Setup: n-blocks world, stacking goal via $goal$ atoms; abstraction inspired by GN1/GN2 [Slaney and Thiébaux2001]
- States are sets of $on/goal$ atoms, actions like $move(2, 1)$ are single atoms
- P_B derives $clear$, $above$, and $incomplete(L, B)$: an incomplete tower (final position) located in L and topped by B, still missing blocks
- Example: constructive move of N onto T at $\hat{s}$

  $$\hat{s} = incomplete(_, T), goal(N, T), clear(T), clear(N); \quad \hat{a} = move(N, T)$$

- Six abstract states partition by what blocks progress + goal state
  - one $(\hat{s}, \hat{a})$ covers many (s, a) pairs (huge search space reduction)
- Trade-off: abstract space constant in n, but does *not* preserve the optimal policy (NP-hard problem)

Experiments



- Questions — abstract Q-learning gives up formal guarantees:
  - Q1 Stability: convergence not guaranteed — does learning stabilize in practice?
  - Q2 Quality: optimality not guaranteed — are policies still good?
  - Q3 Sample efficiency: abstraction promise — fewer samples?
- 20-blocks world ($\Omega(20!)$ states): abstract Q-learning climbs to baseline (62, naive policy) in ∼500-2000 episodes, concrete Q-learning never leaves it
- MiniGrid DoorKey: similar pattern
- Findings (with further MiniGrid games):
  - consistently high returns after initial learning → Q1;
  - acceptable-quality episodes learned consistently → Q2;
  - good abstract policies where concrete Q-learning does not get a go → Q3

7. Conclusion

Summary

What we covered:

- RL fundamentals (MDPs, Q-learning, Deep RL) and where symbolic models help
- ASP as a modeling/reasoning layer for RL: shaping, guardrail, fixing (runtime) + abstraction (structural)
- Normative ASP: norms as soft (weak-constraint) obligations, contrary-to-duty conflicts
- Own line of work (KBS@TU Wien):
  - guardrail, shaping [Schmitt2025], [Lopez-Miguel *et al.*2025]
  - policy fixing [Adam and Eiter2025]
  - scalability [Adam *et al.*2026]; Adam and Eiter2026] (streaming, sampling)
  - abstraction [Bankosegger *et al.*2026] (CARCASS in ASP)
- Throughout: ASP's declarative machinery repeatedly as a suitable tool for norms, guarantees, structure plain RL can't express

Outlook

- Technical issues:
  - Scalability — the generic, ever-present challenge
  - Grounding, specifically — attack it e.g. by incremental lazy grounding
  - Infinite/continuous state and action spaces — a hard mismatch with ASP's discrete strength, not just a scale issue
  - Parallelization — multi-shot, incremental, streaming workloads raise their own questions
- Bridging ASP and RL:
  - Interfacing with DNNs, beyond DQfD, Q-value queries
  - Tighter integration of ASP and Deep RL, cf. Kautz's [2022] taxonomy
- RL helping ASP (the *reverse*): comparatively underexplored

> *"RL learns from reward, ASP reasons from structure — a neuro-symbolic combination, but a loosely-coupled one in Kautz's sense; the open challenge is a shared representation both can act on."*

References I

Sebastian Adam and Thomas Eiter. ASP-driven emergency planning for norm violations in reinforcement learning. In *AAAI*, pages 14772–14780. AAAI Press, 2025.

Sebastian Adam and Thomas Eiter. ASP-based probabilistic policy fixing for norm compliant RL. In *IJCAI*, 2026.

Sebastian Adam, Francesco Calimeri, Thomas Eiter, Luca Laboccetta, Simona Perri, and Giorgio Terracina. Emergency planning for norm violations via ASP-based stream reasoning. In *LPNMR*, 2026.

Mario Alviano, Carmine Dodaro, Wolfgang Faber, Nicola Leone, and Francesco Ricca. WASP: A native ASP solver based on constraint learning. In Pedro Cabalar and Tran Cao Son, editors, *Logic Programming and Nonmonotonic Reasoning, 12th International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings*, volume 8148 of *Lecture Notes in Computer Science*, pages 54–66. Springer, 2013.

Marcello Balduccini. Learning and using domain-specific heuristics in ASP solvers. *AI Communications*, 24(2):147–164, 2011.

Rafael Bankosegger, Thomas Eiter, and Johannes Oetsch. Answer-set-programming-based abstractions for reinforcement learning. *Theory and Practice of Logic Programming*, pages 1–18, 2026. First View.

Rafael Bankosegger. ASP-based abstractions for reinforcement learning. Master's thesis, Technische Universität Wien, 2025.

Chitta Baral, Vladik Kreinovich, and Raul Trejo. Computational complexity of planning and approximate planning in the presence of incompleteness. *Artif. Intell.*, 122(1-2):241–267, 2000.

Richard Bellman. *Dynamic programming*. Princeton Univ. Press, Princeton, New Jersey, 1957.

Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, 2011.

Pedro Cabalar, Agata Ciabattoni, and Leendert van der Torre. Deontic equilibrium logic with explicit negation. In *JELIA*, volume 14281 of *Lecture Notes in Computer Science*, pages 498–514. Springer, 2023.

Elias Datler. Balancing learning and planning: A case study on combining reinforcement learning with answer set programming in the blocks world. Bachelor's Thesis, Technische Universität Wien, 2020.

References II

Thiago Freitas dos Santos, Paulo E. Santos, Leonardo Anjoletto Ferreira, Reinaldo A. C. Bianchi, and Pedro Cabalar. Heuristics, answer set programming and markov decision process for solving a set of spatial puzzles. *Appl. Intell.*, 52(4):4488–4510, 2022.

Thomas Eiter, Giovambattista Ianni, and Thomas Krennwallner. Answer set programming: A primer. In Sergio Tessaris, Enrico Franconi, Thomas Eiter, Claudio Gutierrez, Siegfried Handschuh, Marie-Christine Rousset, and Renate A. Schmidt, editors, *Reasoning Web. Semantic Technologies for Information Systems, 5th International Summer School 2009, Brixen-Bressanone, Italy, August 30 - September 4, 2009, Tutorial Lectures*, volume 5689 of *Lecture Notes in Computer Science*, pages 40–110. Springer, 2009.

Marc Etheve, Zacharie Alès, Côme Bissuel, Olivier Juan, and Safia Kedad-Sidhoum. Reinforcement learning for variable selection in a branch and bound algorithm. In *CPAIOR*, volume 12296 of *Lecture Notes in Computer Science*, pages 176–185. Springer, 2020.

Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Clingo = ASP + Control: Preliminary Report, May 2014.

Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Multi-shot ASP solving with clingo. *Theory Pract. Log. Program.*, 19(1):27–82, 2019.

Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, pages 1070–1080. MIT Press, 1988.

Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Gener. Comput.*, 9(3/4):365–386, 1991.

Laura Giordano, Alberto Martelli, and Daniele Theseider Dupré. Temporal deontic action logic for the verification of compliance to norms in ASP. In Enrico Francesconi and Bart Verheij, editors, *International Conference on Artificial Intelligence and Law, ICAIL 2013*, pages 53–62. ACM, 2013.

Ricardo Gonçalves and José Júlio Alferes. Deontic logic programs. In *AAMAS*, pages 1333–1334. IFAAMAS, 2013.

Guido Governatori. An ASP implementation of defeasible deontic logic. *Künstliche Intell.*, 38(1-2):79–88, 2024.

References III

Christian Hatschka, Agata Ciabattoni, and Thomas Eiter. Deontic Paradoxes in ASP with Weak Constraints. *Electronic Proceedings in Theoretical Computer Science*, 385:367–380, September 2023.

Christian Hatschka. Representing normative reasoning in answer set programming using weak constraints. Master's thesis, Technische Universität Wien, 2022.

Todd Hester, Matej Vecerík, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, Gabriel Dulac-Arnold, John P. Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep Q-learning from demonstrations. In *Proc. of AAAI 2018*, pages 3223–3230. AAAI Press, 2018.

Wassily Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

B. D. Jovanovic and P. S. Levy. A Look at the Rule of Three. *The American Statistician*, 51:137–139, 1997.

Henry A. Kautz. The Third AI Summer: AAAI Robert S. Engelmore Memorial Lecture. *AI Mag.*, 43(1):93–104, 2022.

Bettina Könighofer, Roderick Bloem, Nils Jansen, Sebastian Junges, and Stefan Pranger. Shields for safe reinforcement learning. *Commun. ACM*, 68(11):80–90, 2025.

Luca Laboccetta, Elena Mastria, Francesco Calimeri, Nicola Leone, Simona Perri, and Giorgio Terracina. Towards effective asp-based stream reasoning: Facilitate the reasoning over patterns of events. In *Proceedings of the 26th International Symposium on Principles and Practice of Declarative Programming*, PPDP '24, New York, NY, USA, 2024. Association for Computing Machinery.

Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV System for Knowledge Representation and Reasoning. *ACM Transactions on Computational Logic*, 7(3):499–562, July 2006.

Matteo Leonetti, Luca Iocchi, and Peter Stone. A synthesis of automated planning and reinforcement learning for efficient, robust decision-making. *Artif. Intell.*, 241:103–130, 2016.

Jia Hui Liang, Vijay Ganesh, Pascal Poupart, and Krzysztof Czarnecki. Learning rate based branching heuristic for SAT solvers. In *SAT*, volume 9710 of *Lecture Notes in Computer Science*, pages 123–140. Springer, 2016.

Vladimir Lifschitz. *Answer Set Programming*. Springer, 2019.

References IV

Ignacio D. Lopez-Miguel, Sebastian Adam, Ezio Bartocci, Thomas Eiter, and Martin Tappler. OFTEN-DeepRL: On-the-fly teaching of ethical norms to deep reinforcement learning agents. In *ECAI*, 2025.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

Matthias Nickles. Integrating relational reinforcement learning with reasoning about actions and change. In Stephen H. Muggleton, Alireza Tamaddoni-Nezhad, and Francesca A. Lisi, editors, *Inductive Logic Programming - 21st International Conference, ILP 2011, Windsor Great Park, UK, July 31 - August 3, 2011, Revised Selected Papers*, volume 7207 of *Lecture Notes in Computer Science*, pages 255–269. Springer, 2011.

Matthias Nickles. A system for the use of answer set programming in reinforcement learning. In Luis Fariñas del Cerro, Andreas Herzig, and Jérôme Mengin, editors, *Logics in Artificial Intelligence - 13th European Conference, JELIA 2012, Toulouse, France, September 26-28, 2012. Proceedings*, volume 7519 of *Lecture Notes in Computer Science*, pages 488–491. Springer, 2012.

Henry Prakken and Marek J. Sergot. Contrary-to-duty obligations. *Stud Logica*, 57(1):91–115, 1996.

Jussi Rintanen. Complexity of planning with partial observability. In *ICAPS*, pages 345–354. AAAI, 2004.

Adrian Schmitt. Deontic challenges combined with ASP-planning on RL-training for the FrozenLake and extensions. Master's thesis, Technische Universität Wien, 2025.

Zohreh Shams, Marina De Vos, Julian A. Padget, and Wamberto Weber Vasconcelos. Practical reasoning with norms for autonomous software agents. *Eng. Appl. Artif. Intell.*, 65:388–399, 2017.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nat.*, 529(7587):484–489, 2016.

References V

John K. Slaney and Sylvie Thiébaux. Blocks world revisited. *Artif. Intell.*, 125(1-2):119–153, 2001.

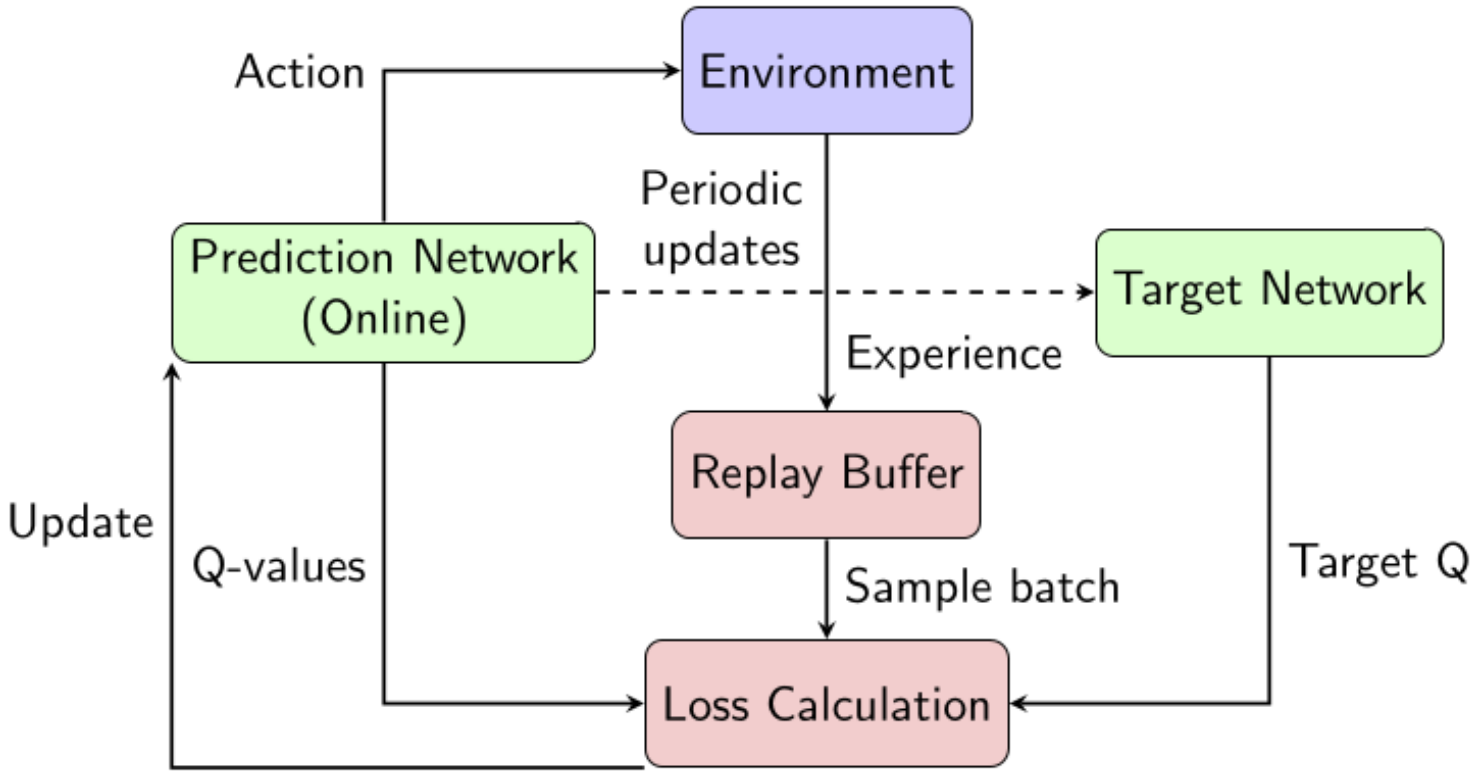
Davide Soldà, Pedro Cabalar, Agata Ciabattoni, and Emery A. Neufeld. Tackling temporal deontic challenges with equilibrium logic. In *AAMAS*, pages 1950–1958. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2025.

Martijn van Otterlo. *The logic of adaptive behavior: Knowledge representation and algorithms for adaptive sequential decision making under uncertainty in first-order and relational domains*, volume 192. IOS Press, 2009.

Christopher J. C. H. Watkins and Peter Dayan. Technical note q-learning. *Mach. Learn.*, 8:279–292, 1992.

Backup Slides

Backup: DQN Mechanics



Backup: DQN Mechanics

- Experience replay:

  store transitions (s, a, r, s') in a replay buffer; train on randomly sampled minibatches
  - breaks correlation between consecutive samples — i.i.d.-like training data
  - reuses past experience, improves sample efficiency
- Target network:

  a second, slowly-updated copy Q_{θ^-} of the network, used only to compute the update target

$$y = r + \gamma \max_{a'} Q_{\theta^-}(s', a')$$

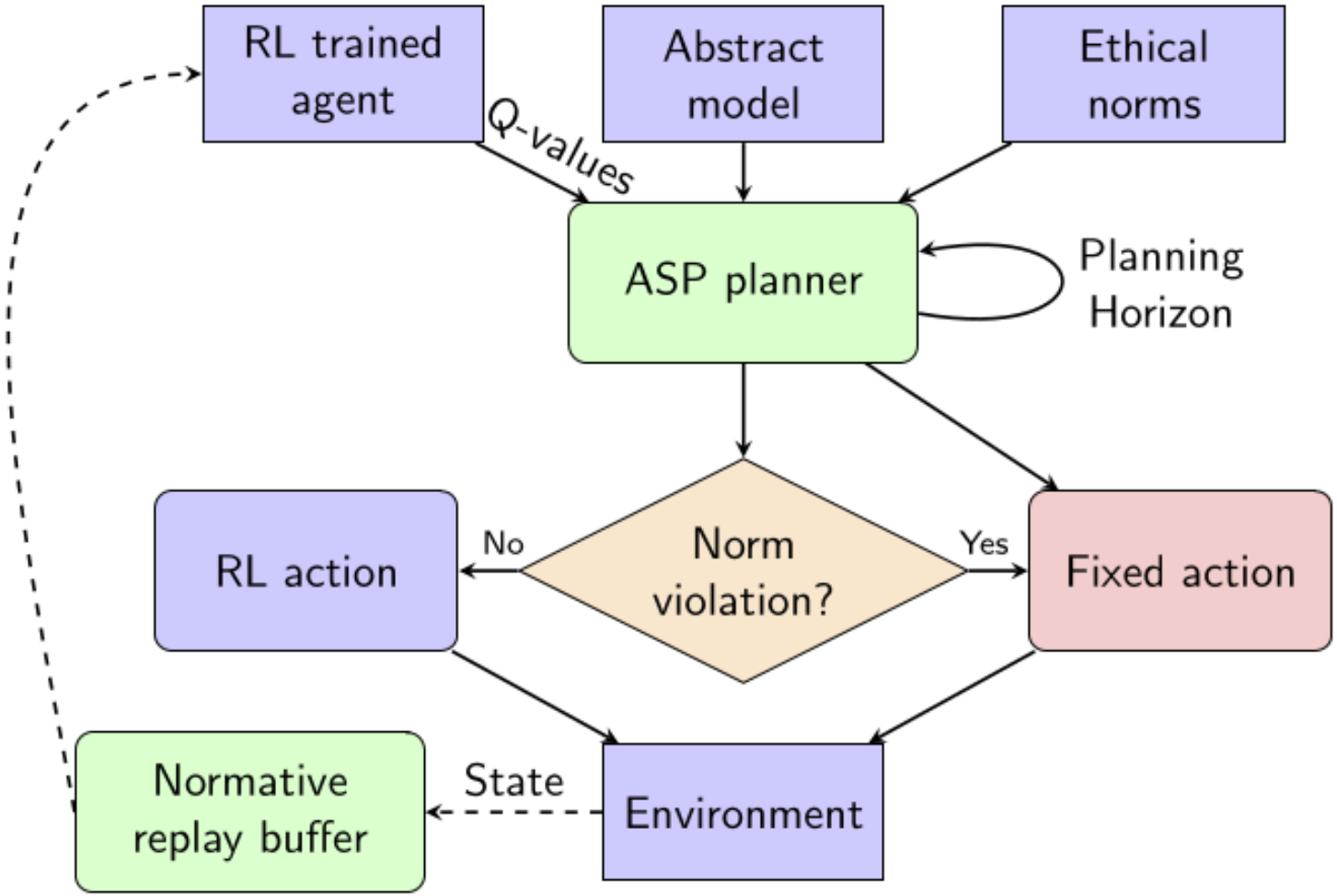
  - θ^- periodically synced to θ (or Polyak-averaged)
  - stabilizes training — without it, the target moves every step, chasing itself
- Together: the two components that made deep Q-learning practically trainable [Mnih *et al.*2015]

Backup: DQfD Expert Loss (J_E)

$$J_E(Q,N) = \frac{1}{|N|} \sum_{(s,a_E,r,s') \in N} \left[\max_{a \in A} \left[Q(s,a) + l(a_E,a) \right] - Q(s,a_E) \right]$$

- a_E: demonstrated (norm-compliant) action from the policy fix, for state s
- $l(a_E,a)$: margin function — 0 if $a = a_E$, else constant $n_c > 0$ (the norm cost)
- Effect: penalizes any $a \neq a_E$ whose Q-value is not at least n_c below $Q(s,a_E)$ — forces a margin favoring the demonstrated action
- n_c task-dependent — set relative to reward scale, so preferring a non-compliant action is meaningfully "punished" [Hester *et al.*2018]

Backup: Policy Fixing for Demonstrations



Backup: The Mud-Water Norm

- Deadline/achievement norm: $dirty \rightarrow O_{\leq K}(reach_water)$

  stepping on mud makes the agent "dirty"; must reach water within K steps or violate the norm
- Encoding mirrors Oil-Fire's activation/persistence pattern (`dirty(T)`, propagated via `in{1}` while within the deadline)
- Distinguishing feature — dynamic policy switch:
  - while `is_dirty` holds, the action-scoring rule (normally driven by the RL policy's preferences) is swapped for a water-seeking policy instead
  - the action predicate stays the same, only the source of its preference scores changes
- If water isn't reached within K steps, a high-priority weak constraint penalizes the violation
- Illustrates DP-sr's modularity: switching behavior in response to a temporal norm requires no change to the rest of the encoding — just which policy feeds the scores [Adam *et al.*2026]

Backup: Probabilistic Policy Fixing — Algorithm Detail

Algorithm (Adam & Eiter)
Input: state s, policy π_Q, confidence $1-\delta$, error tol. ϵ, horizon h.
Output: probabilistic policy fix π'

```
1: compute n (RoT), N (Hoeffding) from delta, epsilon
2: pi'' <- pi_Q ;  c <- 0 ;  Pi <- {}
3: while c*n < N and pi'' not in Pi do
4:     Pi <- Pi u {pi''} ;  c <- c + 1
5:     v <- bad worlds from RoT-Check on pi'' (n fresh samples)
6:     if v = {} then return pi''
// survives the check
7:     else pi'' <- ASP-Gen(pi_Q, bad-world samples so far u v)
8: end while
9: pi'' <- ASP-Gen(pi_Q, full N-sample batch)   // fallback
10: return pi''
```

- Three stopping conditions:
  (1) a candidate survives RoT-Check;
  (2) ASP-Gen repeats a previously tried policy — converged, continuing is futile;
  (3) sample budget N exhausted — fall back to the full-N answer, guaranteed within ϵ of optimal
- RoT-Check is cheap (NP), used repeatedly; ASP-Gen invoked sparingly, folding in more evidence each time [Adam and Eiter2026]

Backup: The Full CARCASS-to-ASP Encoding

Given $\hat{C} = \langle(\hat{s}_1; \hat{A}_{\hat{s}_1}), \ldots, (\hat{s}_n; \hat{A}_{\hat{s}_n})\rangle$, with $\hat{s}_i$ having free variables $\bar{V}$:

- $P_{\hat{s}_i}$ — covering relation for the abstract state:

$$sIdx(\lambda(\hat{s}_i), i). \qquad sCov(\lambda(\hat{s}_i), (\bar{V})) \leftarrow \hat{s}_i.$$

- $P_{\hat{s}_i, \hat{a}_{i,j}}$ — covering relation for $\hat{a}_{i,j} = p(\bar{u})$:

$$aCov(\lambda(\hat{s}_i, \hat{a}_{i,j}), p(\bar{u})) \leftarrow sChoice(\lambda(\hat{s}_i)),\ sCov(\lambda(\hat{s}_i), (\bar{V})),\ p(\bar{u}).$$

- P_{Cov} — union over all i, j, plus the choice rule ("guess"):

$$P_{Cov} = \bigcup_i \left(P_{\hat{s}_i} \cup \bigcup_j P_{\hat{s}_i, \hat{a}_{i,j}}\right) \cup \{\, 1 = \{sChoice(R) : sCov(R, _)\}. \,\}$$

- $P_{\hat{C}}$ — add the index-minimality constraint ("check"):

$$P_{\hat{C}} = P_{Cov} \cup \{\, \leftarrow sChoice(R_1), sCov(R_2, _), sIdx(R_1, I_1), sIdx(R_2, I_2), I_2 < I_1. \,\}$$

Backup: CARCASS-to-ASP — Worked Example and Correctness

Abstract state $\hat{s}_2$ (all blocks on the table), action $\hat{a}_{2,1} = move(A, B)$, labels $\lambda(\hat{s}_2) =$ 3towers, $\lambda(\hat{s}_2, \hat{a}_{2,1}) =$ 3t-mAB:

$$P_{\hat{s}_2} = \left\{ \begin{array}{l} sIdx(\text{3towers}, 2). \\ sCov(\text{3towers}, (A, B, C)) :- on(A, table), on(B, table), \\ \qquad\qquad\qquad\qquad\qquad\qquad on(C, table), A \neq B, B \neq, A \neq C. \end{array} \right\}$$

$$P_{\hat{s}_2, \hat{a}_{2,1}} = \left\{ \begin{array}{l} aCov(\text{3t-mAB}, move(A, B)) :- \ sChoice(\text{3towers}), \\ \qquad\qquad\qquad\qquad\qquad\qquad sCov(\text{3towers}, (A, B, C)), move(A, B). \end{array} \right\}$$

Correctness [Bankosegger *et al.*2026], Corollary 1: $P_{\hat{C}}$ is provably a right encoding

- $\mathcal{AS}(P_s \cup P_{A_s} \cup P_{Cov})$ corresponds one-to-one with the abstract states covering s;
- each answer set lists exactly the admissible abstract actions and the concrete actions they cover;
- $\mathcal{AS}(P_s \cup P_{A_s} \cup P_{\hat{C}})$ has at most one answer set — the covering state of lowest index, faithfully reproducing decision-list semantics.

Backup: The Dual Side — RL for ASP

- This talk went one direction (ASP helping RL); what about the *reverse*?

  RL improving ASP's own evaluation is comparatively underexplored

- SAT branching:

  Liang et al. [2016]: LRB, variable selection as a multi-armed bandit, outperformed VSIDS

- MIP/Branch-and-Bound:

  Etheve et al. [2020] — RL for variable selection

- ASP itself:

  learning (non-RL) solver heuristics, e.g. Balduccini [2011] and others

- Open question: could RL-style online learning improve branching or grounding heuristics in clingo/clasp the way it did for SAT?