Representation learning to act and plan: Learning general models and policies

Part I

Tutorial, IJCAI 2026
Bremen, Germany, 8/2026

Blai Bonet
Universitat Pompeu Fabra
Barcelona, Spain

Hector Geffner
RWTH Aachen University
Aachen, Germany

Alexander von
HUMBOLDT
STIFTUNG

RWTH AACHEN UNIVERSITY

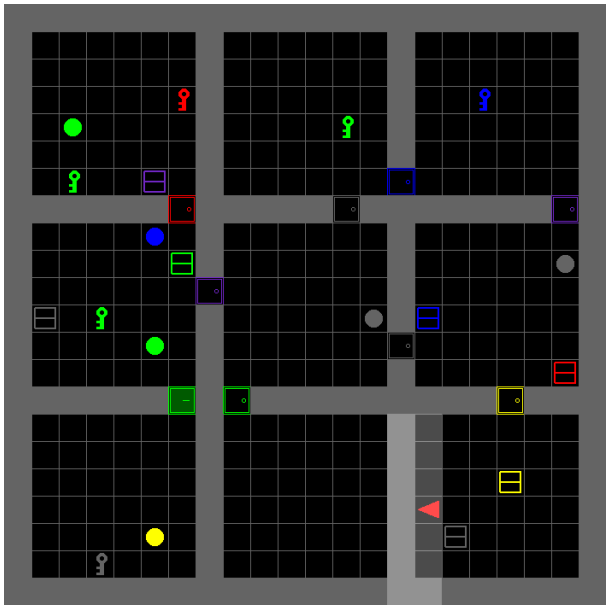
European Research Council
Established by the European Commission

Deep Learning

$$\textit{Input } x \Longrightarrow \boxed{\textsc{Function } f} \Longrightarrow \textit{Output } f(x)$$

- In **deep learning (DL)** and **deep reinforcement learning (DRL)**, training results in function f_θ
- Function f_θ given by structure of **neural network** and adjustable parameters θ
  - ▷ In DL, **input** x may be an image and **output** $f_\theta(x)$ a classification label
  - ▷ In DRL, **input** x may be state of game, and **output** $f_\theta(x)$, value of state
- Parameters θ learned by **minimizing error function** by stoch. gradient descent
- **A true revolution in AI** and out; still **unfolding**
- **Limitations:** OOD generalization, systematic reasoning, ..., **methodology**

Deep Reinforcement Learning (DRL)



- **Task:** *Pick up grey box behind you, then go to grey key and open door*
- Red triangle is agent at bottom right. Light-grey is field of view
- Learn **controller** that accepts **goals** and **obs**, and outputs **action** to do
- Like a "classical planning" **but** state representation/dynamics **not known**, and goals to be achieved **reactively** (not by planning) with policies that **generalize**

Bottom Up Representation Learning

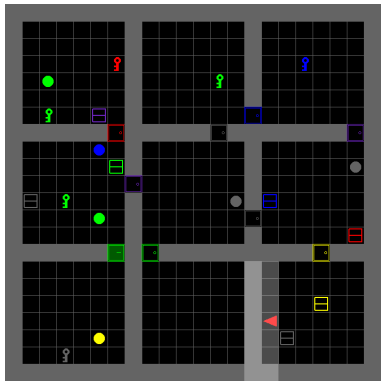
- Surprise is not that DRL methods struggle in Minigrid, but that they manage to generate meaningful behavior at all, given **so little prior knowledge**

- Yet **methodology** largely **ad hoc:** from intuitions to **architectures** and **experiments** using baselines; performance improvements but **no crisp understanding**

- What is **meaning** of *"80% success rate"* when *"100% success rate"* feasible?

  - What is being done "right"? What "wrong"?
  - Missed opportunities for (**human**) learning
  - Condemned to mindless (**human**) exploration

Top-Down Representation Learning



- **The agenda:**
  - ▷ understand what needs to be learned
  - ▷ understand how to learn it; eventually **deep nets** and **SGD**
- **Key questions** for learning to act and plan:
  - ▷ What **languages** for representing general dynamics?
  - ▷ What **languages** for representing general policies, subgoals, hierarchies?
  - ▷ How **representations** over such languages learned?

Key Dimensions: A Challenge Matrix 1/2

Learning what?	Symbolic Methods	Deep Learning
1. Lifted action models	●	●
2. General policies	●	●
3. General subgoaling	●	●

- **Lifted action models:** support infinite # of **instances** (init, goal, # of objects)
- **General policies:** for solving **all instances** in a **domain**
- **General subgoaling:** for decomposing problems into simpler subproblems

Challenge Matrix 2/2

Learning what?	Symbolic Methods	Deep Learning
1. Lifted action models	●	●
2. General policies	●	●
3. General subgoaling	●	●

- This is all **lifted-model-based RL** in **classical planning setting**
  - ▷ *World models, state abstraction, temporal abst, exploration:* they're all here
  - ▷ *Relational state representations* given or learned, key for *generalization*

- Why **care** about **classical planning setting**?
  - ▷ If RL doesn't work robustly here, it won't work robustly elsewhere
  - ▷ Existing RL methods do not actually work well in this setting!
  - ▷ Crisp problem, crisp challenges, crisp (human!) learning

Structure of the Tutorial

- **Part I. About Problem #1:** Learning lifted action models (**Hector**)
- **Part II. About Problem #2:** Learning general policies (**Blai**)
- (**Problem #3:** Learning sketches/hierarchies (involves *temporal abstraction*))

Two approaches to problems #1 and #2: **symbolic** and **deep learning**.

We'll skip #3.

Review: Language of Classical Planning 1/2

- **States and goals** given by sets of atoms $p(o_1, \ldots, o_k)$
  - ▷ atom q true in state s iff it's in the state; $q \in s$

- **Actions** modeled with three *sets of atoms*:
  - ▷ **Add:** atoms that become true
  - ▷ **Del:** atoms that become false
  - ▷ **Precs:** atoms that must be true for action to be applicable

- Resulting language known as **STRIPS**.

- **Extensions** in PDDL: *conditional effects*, *functions*, *defined predicates*, ...

Language of classical planning 2/2

- A planning **domain** D is given by set of **action schemas**
  - *Move*(x, x'): **Prec:** $at(x)$, $adj(x, x')$
    Eff: $at(x')$, $\neg at(x)$
  - *Pick*(o, x): **Prec:** $at(x)$, $at2(o, x)$, $freehand$
    Eff: $hold(o)$, $\neg at2(o, x)$, $\neg freehand$
  - *Drop*(o, x): **Prec:** $at(x)$, $hold(o)$
    Eff: $at2(o, x)$, $freehand$
- An **instance** $P = \langle D, I \rangle$ given by D and instance info $I = \langle Objs, Init, Goal \rangle$
- A **plan** for P is sequence of ground actions that map initial state into goal state

Why classical planning is nice setting for neuro-symb learning?

- Crisp, meaningful, and non-trivial form of **reasoning**
- Perfect testbed for **structural generalization**
  - Infinite number of **instances** $P = \langle I, D \rangle$ over **same domain** D
  - Play with sampled instances and learn about whole domain D
- Crisp, meaningful, and non-trivial **learning problems**:
  - **Learn general strategy** for solving all instances in a given domain (**#2**)
  - **Learn the domain** D itself, including predicates and action schemas (**#1**)

Problem #1: Learning predicates and action schemas

- Consider following **action trace** (executable action seq) in "Delivery":

$$\tau : pick(o_1, c), move(c, c'), drop(o_1, c'), pick(o_1, c')$$

- **Task:** learn hidden domain D from traces such as this. This involves:
  - ▷ learning the **predicates** $at(c)$, $at2(p, c)$, $hold(p)$, . . .
  - ▷ learning the **action schemas** with their effects and preconditions

- Crisp and meaningful problem:
  - ▷ LOCM (Cresswell and Gregory, 2011): heuristic; not sound or complete
  - ▷ SAT (Bonet and G. 2020): uses less input info but not scalable
  - ▷ **SIFT** (Gösgens, Jansen, G. 2025): sound, complete, scalable

Terminology and Assumptions in SIFT

Assumption The hidden STRIPS domain D (with negation) is **well-formed**: i.e., if an action makes a literal true, the complement of the literal is true when the action is executed (invariant or action precondition). This means action effects are effective and change the state.

- **Action trace** in $P = \langle D, I \rangle$: seq. of applicable ground actions from initial state
- **Domain traces:** action traces from instances $P = \langle D, I \rangle$ of D
- **Extended traces:** set of (action) traces that "intersect"; i.e., that have some (hidden) states in common

Dual Representation of Planning Domains

- **Action effects** in a domain expressed as (**sign** of effects left out)
  - ▷ $pick(x_1, x_2)$: has effects over $at2(x_1, x_2)$, $hold(x_1)$, and $handempty$
  - ▷ $drop(x_1, x_2)$: has effects over $at2(x_1, x_2)$, $hold(x_1)$, and $handempty$
- **Alternatively**, the atoms **affected** by actions expressed:
  - ▷ $hold(x_1)$ **affected** by actions $pick(x_1, \_)$ and $drop(x_1, \_)$
  - ▷ $at2(x_1, x_2)$ **affected** by actions $pick(x_1, x_2)$ and $drop(x_1, x_2)$
- If we drop variable names x_i and keep indices, this simplifies to:
  - ▷ $hold$ (only) affected by action patterns $pick[1]$ and $drop[1]$
  - ▷ $at2$ (only) affected by action patterns $pick[1, 2]$ and $drop[1, 2]$
- This notation is very convenient for **learning predicates**; indeed
  - ▷ A **predicate** is "just" the set of action patterns that affect it (**features**)
  - ▷ There is a **finite number** of such features
  - ▷ Only those which are **consistent** with the traces encode domain predicates
  - ▷ They can be found **efficiently**, one by one!

Feature Consistency

- A **feature** f is a pair $f = \langle k, B \rangle$, where B is a set of action patterns of arity k.
  - e.g., $f = \langle 2, \{a[1,3], b[3,2]\} \rangle$ says $f(x_1, x_3)$ effect of $a(x_1, x_2, x_3)$; $f(x_3, x_2)$ effect of $b(x_1, x_2, x_3)$
- A feature $f = \langle k, B \rangle$ is **consistent** with a set of action traces T iff
  - **consecutive patterns** $a[t]$ and $b[t']$ from B in the traces can be assigned different signs

  where $a[t]$ and $b[t']$ from B are **consecutive** in T if two ground actions $a(o_1)$ and $b(o_2)$ in a trace $t \in T$ affect same "hypothetical" atom $f(o)$, i.e., $o = t[o_1] = t'[o_2]$, and **no action between them** affects $f(o)$.
- **Determining** if f is consistent with T is **poly-time**, and reduces to **2-CNF SAT**
  - Formula $sign(a[t]) \neq sign(b[t'])$ for consecutive pairs $a[t]$, $b[t']$ in some trace
- **Theorem (Informal):** Max domain **equivalent** to hidden domain recovered from features consistent with a sufficiently rich set of traces T.

Experimental Results: SIFT (Gösgens, Jansen, G. 2025)

			Full Graphs				Partial Graphs				**Traces**			
Domain	$\#P$	$\#F$	$\#F_a$	$\#E$	Time	Verif	$\#F_a$	$\#E$	Time	Verif	$\#F_a$	$\#E$	Time	Verif
blocks3	3	1k	5.0	21k	51	100%	5.0	81	28	100%	5.0	65	29	100%
blocks4	5	93	9.0	186k	587	100%	9.0	86	17	100%	9.0	85	17	100%
delivery	3	62	5.0	57k	183	100%	5.0	601	105	100%	5.0	350	100	100%
driverlog	4	560	13.0	63k	700	100%	13.0	1k	777	100%	13.0	350	683	100%
ferry	4	31	4.0	15k	347	100%	4.0	251	31	100%	4.0	170	24	100%
grid	6	290	7.0	99k	546	100%	7.0	10k	37	100%	29.7	10k	36	**0%**
grid2	6	1k	7.0	152k	$1k$	100%	7.0	500	123	100%	7.0	800	117	100%
gripper	4	43	6.0	95k	212	100%	6.0	230	64	100%	6.0	250	306	100%
hanoi	2	134	4.0	59k	$2k$	100%	4.0	50	25	100%	4.0	25	24	100%
logistics	2	212	7.0	648k	$12k$	100%	7.0	5k	1403	100%	7.0	350	1304	100%
miconic	3	99	8.0	127k	376	100%	8.0	46	33	100%	8.0	60	36	100%
npuzzle	2	912	26.0	483k	$11k$	100%	26.0	130	312	100%	26.0	200	311	100%
sokoban	2	352	3.0	26k	231	100%	3.0	10k	195	100%	126.2	10k	220	**4%**
sokoban2	2	20k	3.0	66k	401	100%	3.0	300	77	100%	3.0	300	123	100%

Number of objects $\#O$ in training instance, number of non-static predicates $\#P$, number of features to test $\#F$, and for each input (full and partial graphs, plain traces): avg. number of admissible features $\#F_a$, avg. number of edges in input graphs $\#E$ (trace length for traces), avg. total time in secs (data generation, learning, verification), and ratio of successful verification tests (Verif). Averages over 25 runs except for full graphs

Model learning with SIFT: Discussion

- SIFT learns **state representation** and action schemas from **action traces** alone
  - ▷ Pretty remarkable: state structure implicit in actions and their arguments!
  - ▷ No state observability needed at all
- Yet, not a **lifted** alternative to **model-based RL**
  - ▷ STRIPS actions need **too many arguments**; e.g, n-*sliding-puzzle*
- **Alternative:** trade action arguments by **state observations**
  - ▷ **No action arguments** : L1 (Balyo *et al.* 2024), SYNTH (Jansen, Gösgens, G. 2025;2026), DIAS (Reiter, Gebler, G. 2026)
  - ▷ **Images** in place of states: ROSAME (Xi, Gould, Thiebaux 2024; 2026)
- Yet most approaches learn from symbolic states and full STRIPS actions; noise, occlusions (Le *et al.* 2024; Lamanna *et al.*, 2025); also numerical variables (Mordoch *et al.*, 2026).

How SIFT simplifies when both actions and states observed?

- Predicates p and their arities k **known**, but not how they are **affected**
- Collect state transitions $[s, a(o), s']$ for each **action schema name** a
- Construct **feature** $p = \langle k, B \rangle$ where B is set of a-**action** patterns
- **Consistency** of feature p easily tested over a-**transitions** $[s, a(o), s']$, not **traces**
- **Feature** p **consistent** p if consistent with the observed **atom** p changes
- **Action preconditions** determined in standard way; p[t] precond of $a(x_1, \ldots, x_n)$ if p(o) true when $a(o')$ for $o = t[o']$

Learning Lifted STRIPS From STRIPS+ Actions

- SIFT does not provide by itself a **lifted alternative to model-based RL**
  - ▷ STRIPS actions need **too many arguments**; e.g, $up(t_7, c_5, c_1)$ in n-puzzle
- We consider next learning STRIPS with guarantees from **state-action** traces where **actions** don't spell all arguments
- The problem is cast as learning STRIPS+ models from **state-action** traces where actions are STRIPS+
- STRIPS+ distinguishes **explicit**/**implicit** action args; only former are **observed**
  - ▷ e.g., all action arguments $up(t_7, c_5, c_1)$ become implicit in STRIPS+, where the actions become $up()$, $down()$, $right()$, $left()$

STRIPS+

- Extension of STRIPS where preconditions $\phi(x, y, z)$ involve three variables types:
  - ▷ **explicit** variables $x = \{x_1, \ldots, x_n\}$, $n \geq 0$ appearing as **action arguments**
  - ▷ existential variables $y = \{y_1, \ldots, y_l\}$, $l \geq 0$, **can't appear** in action effects
  - ▷ **determined** variables $z = \{z_1, \ldots, z_m\}$, $m \geq 0$, **can appear** in action effects

- **Determined** action args z_i in first PDDL standard (McDermott *et al.* 1998); declared as `:vars` not `:parameters`

- Determined variables z result from **state-invariants**
  - ▷ In all reachable s where ground $a(o)$ applies, there is a unique grounding for z among the (non-empty) groundings that satisfy $\phi(x, y, z)$ with $x = o$.
    - – Precondition of action $up()$ in puzzle is $atB(z_1)$, $above(z_1, z_2)$, $atT(z_3, z_2)$; i.e., Blank at z_1, z_2 above z_1, and tile z_3 at z_2.

Stratified STRIPS+

- Evaluation of arbitrary STRIPS+ preconditions is **intractable** (they are CSPs)
- **Stratified** STRIPS+ preconds $\phi(x, y, z)$ are **easy to evaluate and learn**
  - Given by $\bigwedge_{i=1,m} Q_i(x, y, z^i)$ where variable z_i only occurs in z^j, $j \geq i$
  - Prefix $Q_1(x, y, z^1)$, . . . , $Q_i(x, y, z^m)$, determines (unique binding for) z_i
- Also, each variable y_i in y appears at most once in $\phi(x, y, z)$
- CSP (constraint satisfaction problem) defined by $\phi(x, y, z)$ in s where $a(o)$ applies
  - has a **unique solution** for variables $z_1, \ldots, z_m$,
  - solution can be found **one variable** z_i at a time, $i = 1, \ldots, m$

SYNTH (Jansen, Gösgens, G. 2025)

Learning Task: Given **state-action** traces $s_0, a_0, \ldots, s_n, a_n$ from hidden **stratified** STRIPS+ domain $P = \langle D, I \rangle$, learn STRIPS+ D_L equivalent to D (same traces).

Algorithm SYNTH

1. Learn **valid query precondition** $\phi(x, y, z)$ that binds the z variables uniquely
   - Construct **stratified** $Q_1(x, y, z^1), \ldots, Q_m(x, y, z^m)$, one **query** Q_i at a time
   - Construct each **query** $Q_i(x, y, z^i)$, one **lifted atom** $q(x, y, z^i)$ at a time
2. Push other **valid** lifted preconditions in $\phi(x, y, z)$
3. Learn the **effects** using x and z variables in standard way

Steps 2 and 3 as when learning from full STRIPS state-action traces; Step 1 computed **greedily**; **sound and complete**

Experimental Results

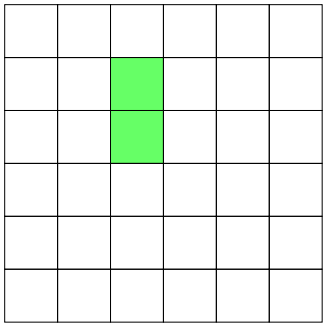
	Data				Learning					Verification			
Domain	#O	#L	$\|x'\|$	$\|x\|$	$\|x\|+\|z\|$	$\|z\|$	$\|x' \setminus z\|$	$\|z \setminus x'\|$	T	$\#\mathrm{O_V}$	$\#\mathrm{S_V}$	$\mathrm{T_V}$	%V
blocks3	5	250	7	5	7	2	0	0	$0.82s$	6	1200	$80.56s$	100%
blocks4	5	250	6	3	6	3	0	0	$1.43s$	6	1600	$105.04s$	100%
delivery	16	1000	9	5	9	4	0	0	$101.4s$	24	1200	$246.69s$	100%
driverlog	63	10000	19	10	19	9	0	0	$1695.32s$	148	2400	$3757.22s$	100%
ferry	8	100	6	2	6	4	0	0	$0.96s$	10	1200	$68.78s$	100%
grid	50	10000	13	4	17	13	0	4	$584.62s$	48	2000	$1264.47s$	100%
gripper	10	500	8	3	11	8	0	3	$2.52s$	12	1000	$96.14s$	100%
hanoi	8	200	3	2	3	1	0	0	$1.5s$	10	400	$41.58s$	100%
logistics	35	1000	13	7	20	13	0	7	$28.25s$	45	1600	$377.5s$	100%
miconic	9	600	8	2	8	6	0	0	$2.78s$	12	1600	$104.14s$	100%
n-puzzle	34	1000	16	0	16	16	0	0	$498.28s$	34	1600	$1123.36s$	100%
c-puzzle	49	500	12	0	12	12	0	0	$147.19s$	49	1600	$569.38s$	100%
sokoban	30	1000	5	2	5	3	0	0	$20.7s$	30	800	$72.45s$	100%
sokopull	25	600	8	3	8	5	0	0	$10.67s$	30	1200	$80.51s$	100%

$\#O$, $\#L$: # of objects, length of trace. $|x'|$ is total # action args in hidden STRIPS domain, $|x|$ is # of action args in resulting hidden STRIPS+ D, and $|z|$ is # of implicit args learned in D_L. T is total time to learn D_L, $\mathrm{T_V}$ is time to verify D_L, . . . , %V is the success rate of the verification.

Summary: Symbolic Methods for Action Model Learning

- We focused on **two settings** and **two algorithms**:
  - ▷ SIFT: learns from **action traces** of hidden STRIPS domain
  - ▷ SYNTH: learns from **state-action traces** of hidden STRIPS+ domain
- Strengths and weaknesses:
  - ▷ SIFT: **invents** predicates but needs "right action arguments" (hidden STRIPS)
  - ▷ SYNTH: builds **referring expressions** for missing args, but needs "right predicates" (hidden STRIPS+)
- Recently, SYNTH+ (Gösgens, Jansen, G. 2026)
  - ▷ Integrates and generalizes both SIFT and SYNTH
  - ▷ SYNTH+ still **incomplete** as approach to **lifted model-based RL** ...
- Skipped over many threads in lifted model learning (noise, occlusion, images, ..)

Example: Showing Limitations of SYNTH



- The dynamics:
  - ▷ Agent is made up of **two green cells**
  - ▷ It can move up, down, right, left; **no arguments**
  - ▷ **State representation:** Atoms $green(cell)$, $right(cell, cell')$, $above(cell, cell')$
- Can SYNTH learn this dynamics?
  - ▷ It can be represented in STRIPS+; e.g., $MoveRight()$:
    - **Precond:** $green(cell), green(cell'), above(cell, cell'), right(cell, c1), right(ce$
    - **Effects:** $green(c1)$, $green(c2)$, $\neg green(cell)$, $\neg green(cell')$
  - ▷ But SYNTH won't learn it; it's **not stratified**.
  - ▷ **Axiom** $TopGreen(cell)$ iff $green(cell) \wedge \neg\exists y.green(y) \wedge above(y, cell)$ OK
  - ▷ But 1) extra predicate to be learned, 2) won't work for **other agent shapes**

Learning STRIPS Models: Deep Learning Approaches

Learning STRIPS Models: Deep Learning Approaches

- Learning from **image sequences**: LATPLAN, ROSAME-2, VISA (Asai, Fukunaga; Xi, Gould, Thiebaux 2026; Rakhman, Schulte, G. 2026)

- Challenging and results mixed; we focus on a **simpler problem**: learning from symbolic traces but using **deep learning**

  ▷ From **action traces** using **transformers** (Muñoz-Molina, Gomez, G. 2026)

  ▷ From **state traces** and **action names** using **GNNs** (Reiter, Gebler, G. 2026)

- Why?

  ▷ **Differentiable learning component** can be integrated into larger architecture

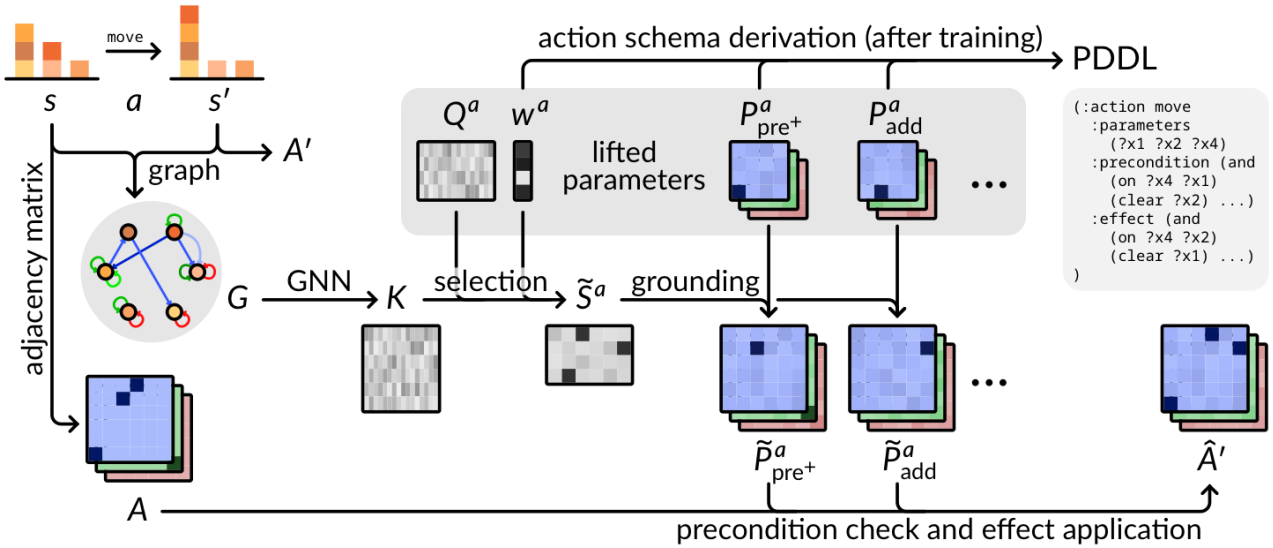
  ▷ **Methodology:**

  – aim at nearly perfect learning or logical reason that prevent it
  – learn to use deep learning tools as another type of solvers
  – embrace of DL not license for arbitrary hacking or law of the jungle!

Learning from State-Action Traces [Reiter, Gebler, G. 2026]

- System learns lifted STRIPS from **state-action** traces/triplets (s, a, s') where a's
  - ▷ Full STRIPS actions: $unstack(a, b)$
  - ▷ STRIPS+ actions; determined arguments omitted: $unstack(a)$
  - ▷ **action names** only: $unstack$
- We focus on last input; **action names only**. **Key challenge:**
  - ▷ Identify objects $o_1, \ldots, o_{k_a}$ in transitions (s, a, s') from **changes** s to s'
  - ▷ Use atoms in s to infer other (determined) arguments of action a
- **Assumptions:**
  - ▷ Like SIFT and SYNTH, hidden STRIPS domain is **well-founded**
  - ▷ No pair of actions $a(o)$ and $a(o')$ account for same state transition (s, s')
  - ▷ State predicates unary or binary
- **Result:** Actions $a(o_1, \ldots, o_{k_a})$ are STRIPS+ actions; other arguments determined. Yet $a()$ not necessarly STRIPS+

Deep Synth: Architecture (DIAS)



- **Graph neural network (GNN)** from $s \cup \Delta(s, s')$ generates embeddings $f(o)$ for the objects o
- **Attention mechanism** binds K of these objects to K action slots
- **Learnable parameters** for encoding lifted action preconditions and effects:
  - ▷ Learnable $w_{ij}^{p,a} \in [0, 1]$: prob that i-th arg of p in a is j-th action arg
  - ▷ Learnable v_j^a: prob that j-th arg of a is active (not all K slots actual action args)
- **STRIPS model extraction**: uses thresholding
- **Loss function:** Predict successor states, maximize valid preconditions, ..

Experimental Results: DIAS (aka of Deep Synth)

	Full STRIPS actions					STRIPS+ actions					Action names only				
Domain	Pr.	Re.	N_s	N_c	N_{sc}	Pr.	Re.	N_s	N_c	N_{sc}	Pr.	Re.	N_s	N_c	N_{sc}
Blocks-3	1.00	1.00	10	10	10	1.00	1.00	10	10	10	1.00	1.00	10	10	10
Delivery	1.00	1.00	10	10	10	1.00	1.00	10	10	10	1.00	1.00	10	10	10
Driverlog	1.00	1.00	10	10	10	1.00	1.00	10	10	10	0.80	0.91	8	8	8
Gripper	1.00	1.00	10	10	10	1.00	1.00	10	10	10	1.00	1.00	10	10	10
Hanoi	1.00	1.00	10	10	10	1.00	1.00	10	10	10	0.81	0.99	8	8	8
Logistics	1.00	1.00	10	10	10	0.82	1.00	7	10	7	0.94	0.97	9	8	7
Miconic	1.00	1.00	10	10	10	1.00	1.00	10	10	10	1.00	1.00	10	10	10
N-Puzzle	1.00	1.00	10	10	10	1.00	1.00	10	10	10	1.00	1.00	10	10	10
Satellite	1.00	1.00	10	10	10	1.00	0.99	1	3	0	0.74	1.00	0	10	0
Sokoban	1.00	1.00	10	10	10	1.00	1.00	10	6	6	1.00	1.00	10	6	6
Sokoban2	1.00	1.00	10	10	10	1.00	1.00	10	10	10	1.00	1.00	10	10	10
Spanner	1.00	1.00	10	10	10	1.00	1.00	10	10	10	1.00	1.00	10	10	10
Visitall	1.00	1.00	10	10	10	1.00	1.00	10	10	10	1.00	1.00	10	10	10

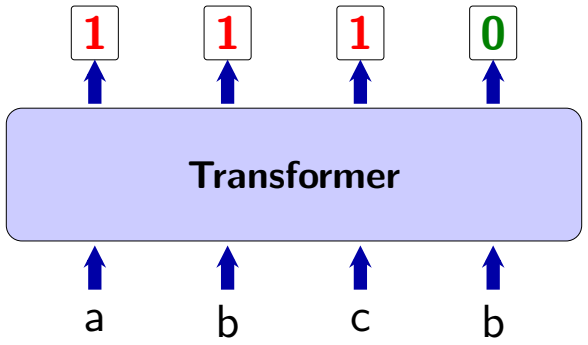
DIAS: 10 runs in each IPC domain, 3 obs. variants. Avg precision (Pr.) and recall (Re.), # of learned domains that are sound, complete, or both as N_s, N_c, and N_{sc}. 12 of the 13 domains learned accurately in most of the 10 runs with **action names only**; 8 learned accurately in **all** runs.

Deep SYNTH/DIAS: Discussion

- **Single domain** not learned accurately in **any** of the 10 runs:
  - ▷ **Satellite** violates assumptions; hidden model not well-founded and multiple $a(o)$ actions account for some transitions

- **DIAS** can be regarded as a **"Deep SYNTH"**; similar limitations/scape
  - ▷ SYNTH can be extended to learn from **action names** even if not STRIPS+
  - ▷ DIAS can't express dynamics of 2-cell agent above either: interestingly,
    - it can identify cells to be updated via GNNs,
    - but it can't express this selection in preconditions, bound to the **given state predicates**

Learning from STRIPS Action Traces using Transformers



Task: **classify** traces (action sequences) as positive (executable) or negative

- **Assumption:** hidden propositional STRIPS model ; token = ground action
- **Data:** random act. sequences; negative if previous action falsifies precondition
- **Architecture:** learnable embeds, no explicit pos encodings, stick-breaking attn
- **Readout:** final token embeddings mapped to Booleans
- **Training:** min of cross-entropy; predict seq so far positive or negative

(Muñoz-Molina, Gomez, G. 2026)

Learning STRIPS Using Transformers: Model Extraction

By including auxiliary actions **init-p** (no preconds; adds p) and **test-p** (precond p, no effect) for each atom p:

- **Symbolic propositional STRIPS** extracted from trained **trace classifier**
  - ▷ **symbolic state** s after action prefix σ determined by applicable test-p actions
  - ▷ propositional action **effects** and **preconditions** easily obtained from them
  - ▷ by exploiting structure of actions $a(o_1, \ldots, o_k)$ model can be **lifted**
- Learned STRIPS model can be used to plan with **planners off-the-shelf**

	Blocksworld			
Model	Train	Test	Plan (M_S)	Plan (M_L)
NoPE, SB	**1.0** (**1.0**)	.998 (.999)	**1.0** (**1.0**)	**1.0** (**1.0**)
SinPE, Softmax	.994 (.998)	.237 (.250)	**1.0** (**1.0**)	.00 (.00)
RoPE, Softmax	.998 (.999)	.363 (.377)	**1.0** (**1.0**)	.00 (.00)

	Npuzzle			
Model	Train	Test	Plan (M_S)	Plan (M_L)
NoPE, SB	**1.0** (**1.0**)	.997 (.997)	**1.0** (**1.0**)	**1.0** (**1.0**)
SinPE, Softmax	.999 (.999)	.253 (.253)	**1.0** (**1.0**)	.01 (.01)
RoPE, Softmax	.998 (.999)	.402 (.445)	**1.0** (**1.0**)	.02 (.03)

	Logistics			
Model	Train	Test	Plan (M_S)	Plan (M_L)
NoPE, SB	**1.0** (**1.0**)	.987 (.993)	**1.0** (**1.0**)	**1.0** (**1.0**)
SinPE, Softmax	.999 (**1.0**)	.281 (.284)	**1.0** (**1.0**)	.00 (.00)
RoPE, Softmax	.999 (.999)	.412 (.402)	**1.0** (**1.0**)	.32 (.06)

	Maze			
Model	Train	Test	Plan (M_S)	Plan (M_L)
NoPE, SB	**1.0** (**1.0**)	.958 (.967)	.98 (.98)	.93 (.93)
SinPE, Softmax	**1.0** (**1.0**)	.510 (.506)	.98 (.98)	.00 (.00)
RoPE, Softmax	**1.0** (**1.0**)	.578 (.562)	.98 (.98)	.00 (.00)

Learning STRIPS Models with Transformers: Discussion

- Stick-breaking attention transformer **classifies** long traces well
- Pure **vanilla transformer** does **not** generalize that well
- Yet **STRIPS models** extracted support nearly-prefect **planning**

Differences with **symbolic SIFT**:

- SIFT learns from **positive** traces only
- SIFT learns **lifted** models and doesn't need the atoms p
- SIFT however needs such atoms p for **planning** with aligned language

Summary: Lifted Action Model Learning

- Symbolic algorithms:
  - ▷ SIFT: **invents** predicates but needs "right" action arguments (hidden STRIPS)
  - ▷ SYNTH: builds **referring expressions** for missing args; needs "right" predicates
- Deep learning:
  - ▷ Deep propositional SIFT: **transformers** to predict next (action) token
  - ▷ Deep SYNTH (DIAS): **GNNs** and form of **slot-attention** to predict next states
- Skipped over other threads in model learning (images, noise/occlusion, numerical)
  - ▷ Xi, Gould, Thiebaux 2024, 2026; Le *et al.* 2024; Lamanna *et al.*, 2025; Mordoch *et al.*, 2026; ...
- **Motivation:** **lifted** alternative to black-box **model-based RL**
- **Next:** Policy learning by Blai ...