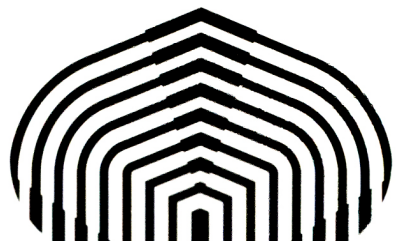
Representation Learning and Synthesis for Generalized Planning

Blai Bonet

Universidad Simón Bolívar, Venezuela

ECAI 2020. August-September 2020

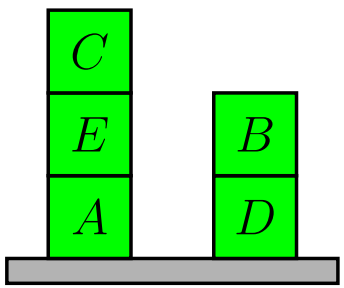


Collaborators

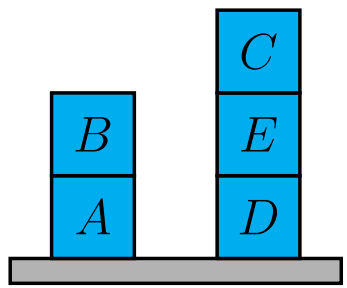
Generalized Planning: The Challenge

Generalized planning is about obtaining a **general plan or strategy** for solving collections of planning problems

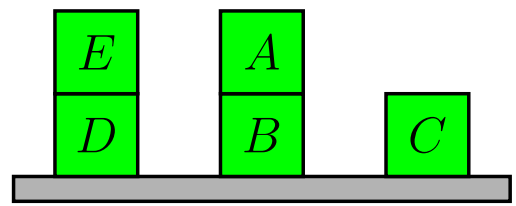
For example, find general strategy to achieve **fixed goal** in all Blocksworld problems, independently of number or initial configuration of blocks



Get A clear



Get A on B



Get tower A, B, C, D, E

Challenges:

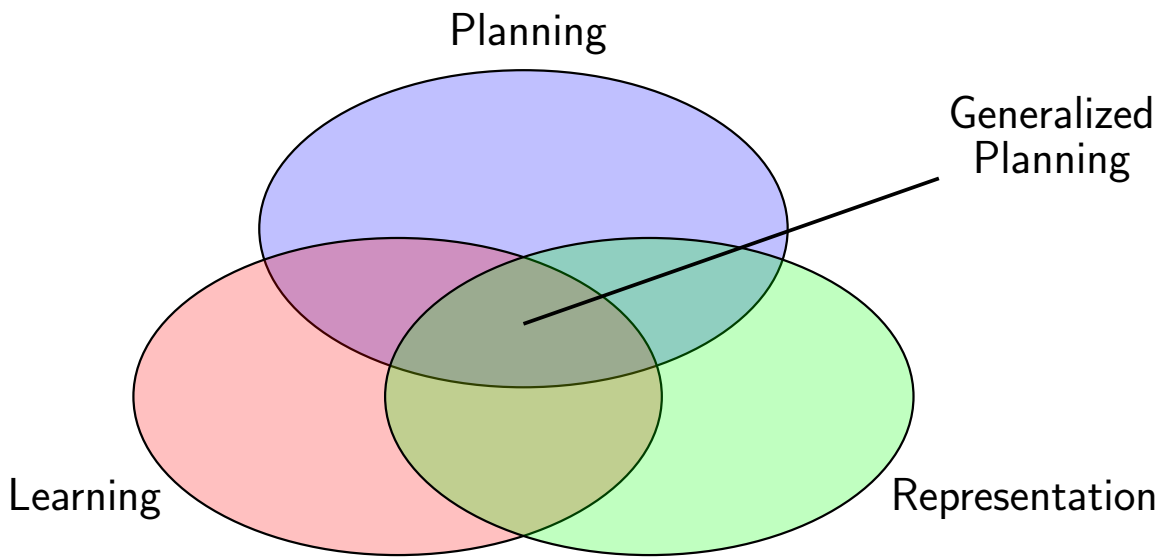
- How to formulate and represent the problem formally?
- What's the form of solutions?
- How to solve it?

Motivation

Why solve individual problems from scratch if it's possible to learn general plan in one shot?

Lots of current research in **deep (reinforcement) learning** is about computation of general plans or policies; e.g. [Espeholt et al., 2018; Groshev et al., 2018; Chevalier-Boisvert et al., 2019; François-Lavet et al. 2019]

Generalized planning gives us a crisp vocabulary to talk about general plans [Levesque, 2005; Hu & De Giacomo, 2011; B. & Geffner, 2015; Belle & Levesque, 2016; Jiménez et al., 2019; Illanes & McIlraith, 2019]

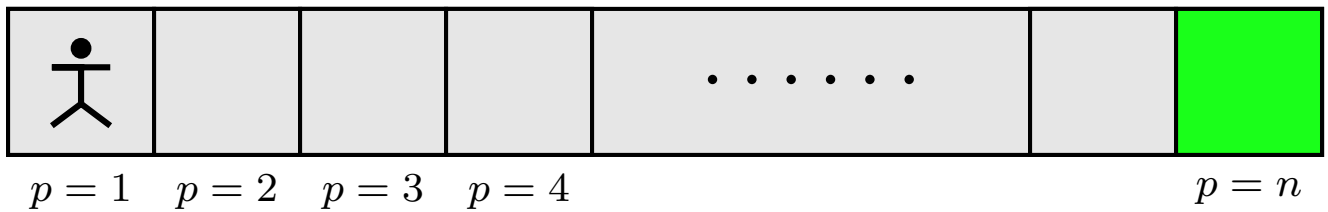


Outline

1. Non deterministic abstractions for families of classical planning problems
2. Qualitative Numerical Planning (QNP): concrete language for abstractions
3. First-order relational planning instances
4. Solving QNPs via reduction to FOND planning
5. Learning QNP abstractions from traces
6. Wrap Up

Example

Problem P_n: agent moves in $1 \times n$ grid, starts at $p = 1$ and goal is $p = n$



Agent can move **right** and **left**, and **senses** R when in last cell

Collection of problems: $\mathcal{Q} = \{P_n : n = 1, 2, \ldots\}$

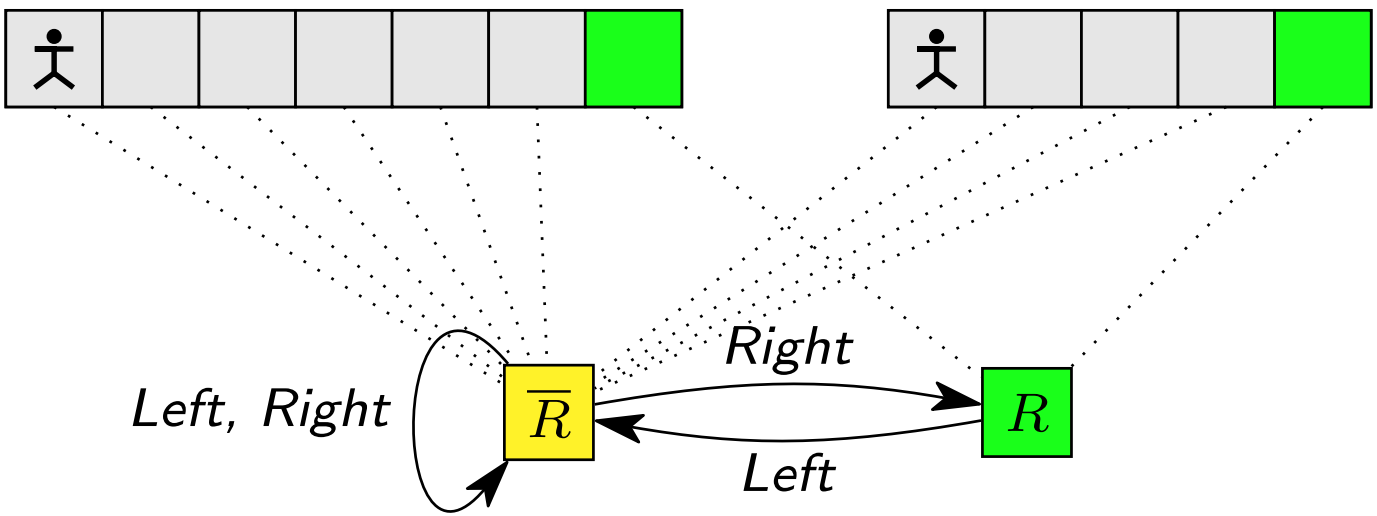
(Generalized) plan π:

While R not sensed do move right

What's the relation between the solution and the collection $\mathcal{Q}$?

Abstraction: Observable Projection

Policy π solves a **single abstract problem** that is **fully observable and non-deterministic (FOND)**



Fairness Assumptions in FOND problems

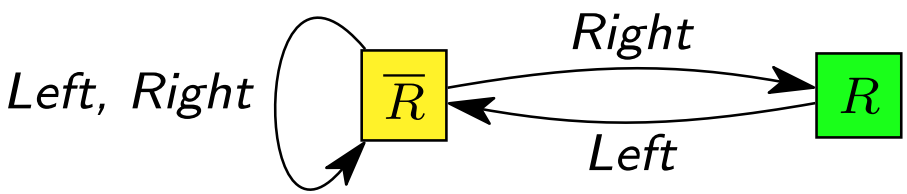
Solutions for FOND problems are either [Cimatti et al., 2003]:

– **Strong solutions:** acyclic and suitable for **adversarial planning**

– **Strong cyclic solutions:** assume **"strong fairness"** and suitable for MDPs when goal is to be reached with probability 1

Fairness assumptions in the projection doesn't correspond to either

Indeed, policy that interleaves *Right* and *Left* solves projection but doesn't solve any P_i as the agent won't get past the second cell

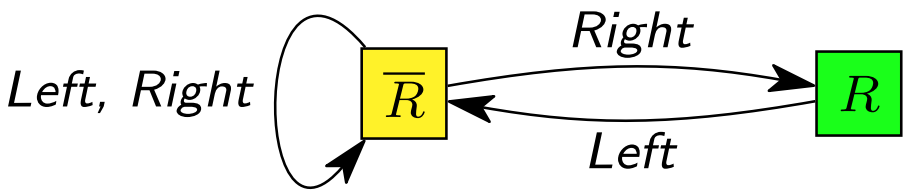


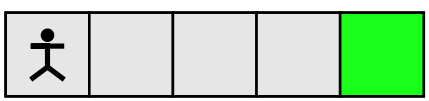
Local vs. Global Properties

Projection captures **local properties** of instances: each transition represents one or more transition in one or more instances

Global properties are lost in projection

Example: there is **no trajectory** (in some P) where *Right* applied infinitely often, *Left* only finitely, and goal isn't reached, but there is such in projection

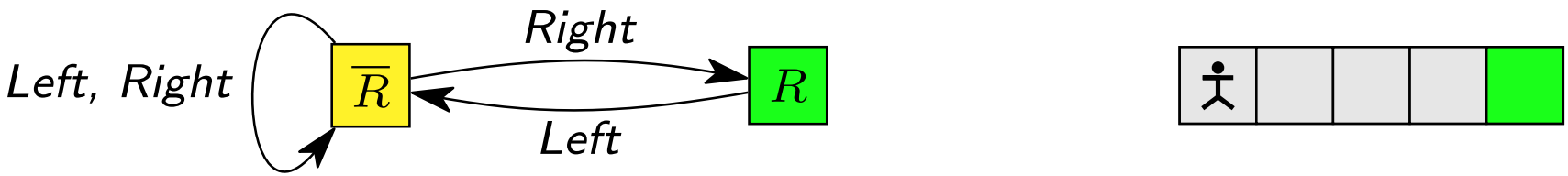




Trajectory Constraints

Global properties can be captured with **trajectory constraints** that filter out non-realizable trajectories [B. et al, 2017]

Example: if *Right* is applied infinitely often and *Left* only a finite number of times, goal is **eventually reached**



Solutions for $\mathcal{Q}$ are **strong-cyclic policies** whose induced trajectories that **comply with the constraints** are goal reaching

Different from standard fairness

Qualitative Numerical Planning

Simple and expressive language [Srivastava et al., 2011]:

- directly express the projection in compact form
- transparently adds the needed trajectory constraints
- after proper reduction to FOND, solved with off-the-shelf FOND planner
- models can be learned from sampled trajectories

A QNP problem is similar to STRIPS problem but extended with **numerical variables** that can be incremented or decremented **qualitatively**

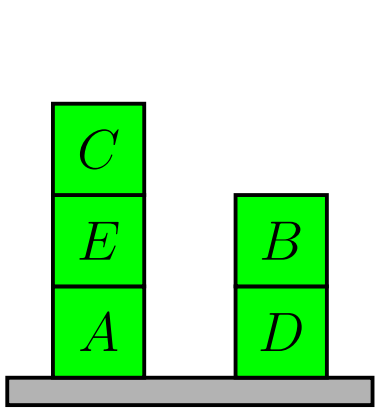
QNP Example: Clear-x

Goal: Remove all blocks above fixed block x

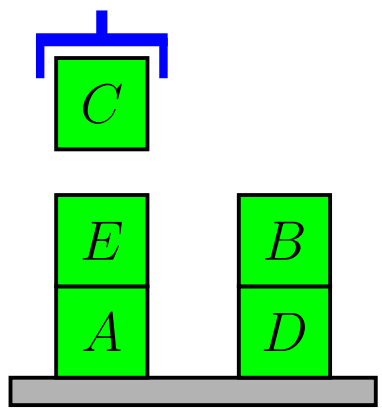
QNP $Q_{clear} = (F, V, I, A, G)$ captures all Blocksworld problems:

- $F = \{H\}$ where H denotes whether gripper holds a block
- $V = \{n\}$ where n **"counts"** blocks above x ($n{>}0$ iff some block above x)
- $I = \{\neg H, n > 0\}$
- $G = \{n{=}0\}$

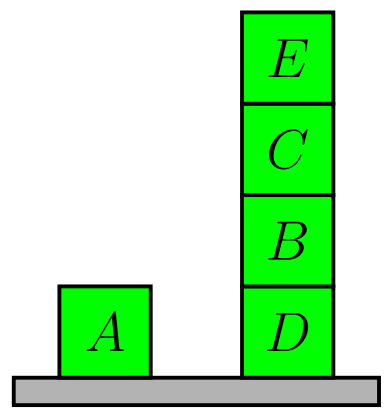
$x =$ block A



$\neg H,\ n = 2$



$H,\ n = 1$

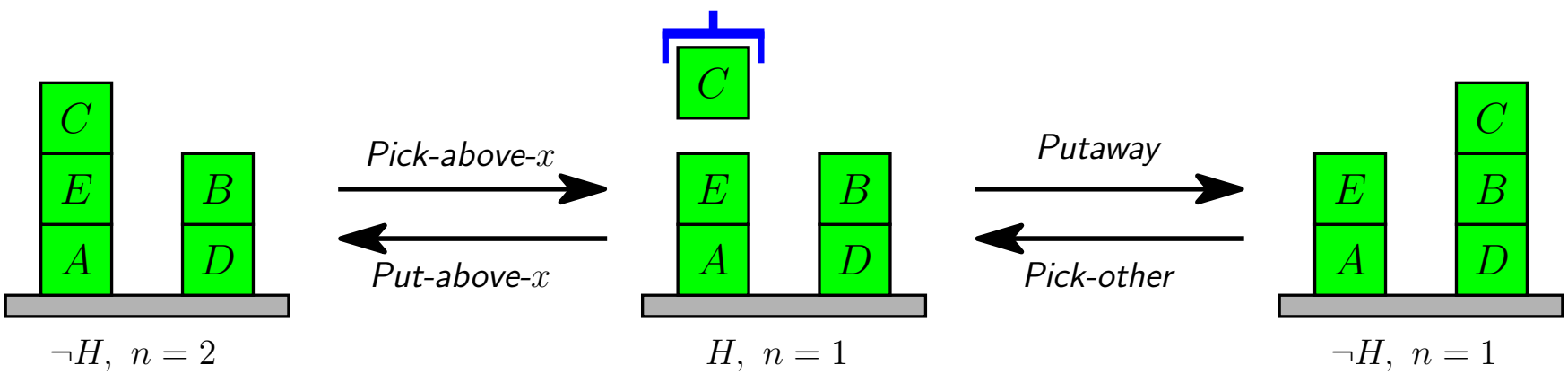


$\neg H,\ n = 0$

QNP Example: Actions

- *Putaway* $= \langle H; \neg H \rangle$ puts held block on table or block not above x
- *Pick-above-*$x = \langle \neg H, n > 0; H, n{\downarrow} \rangle$ picks the top block above x
- *Put-above-*$x = \langle H; \neg H, n{\uparrow} \rangle$ puts block being held on top block above x
- *Pick-other* $= \langle \neg H; H \rangle$ picks block not above x

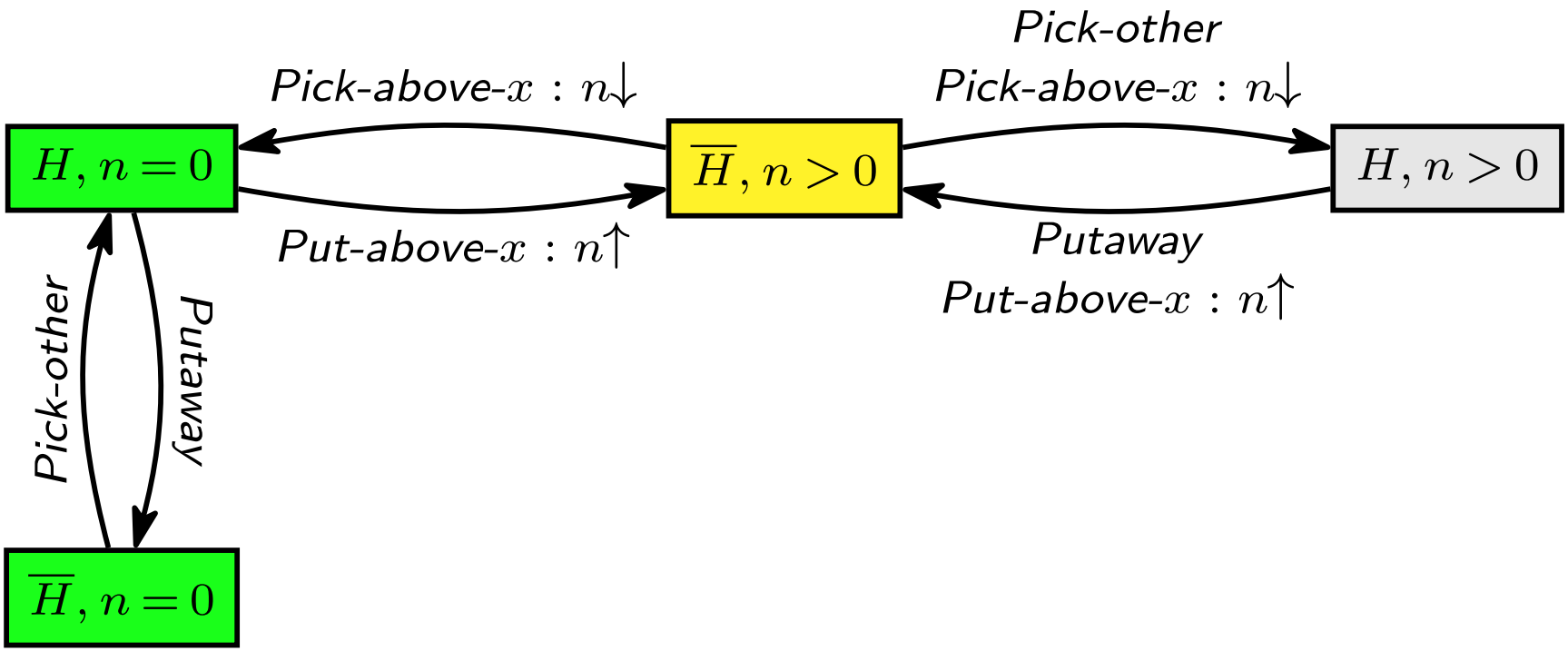
Block x is block A:



QNP Example: Projection

Observation projection for $Q_{clear} = (F, V, I, A, G)$ where

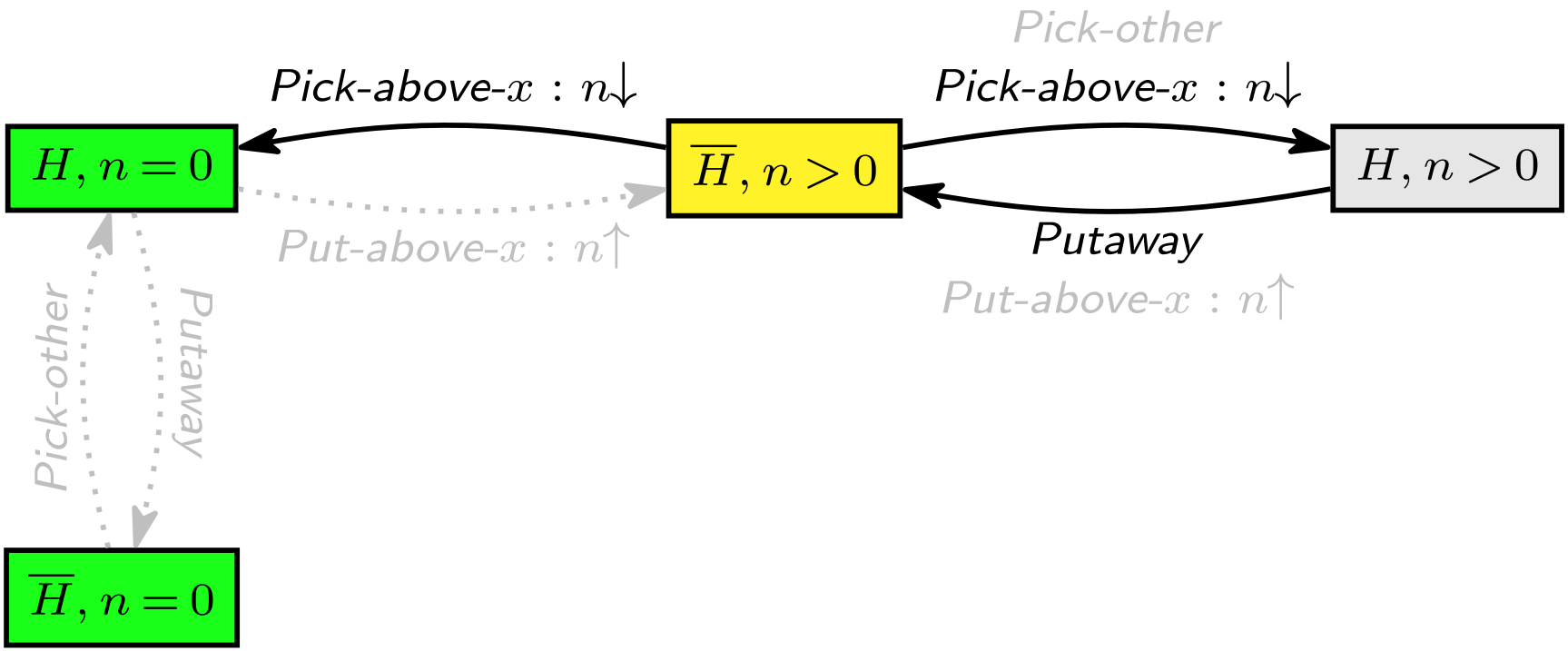
- $F = \{H\}$ and $V = \{n\}$
- $I = \{\overline{H}, n > 0\}$ and $G = \{n = 0\}$
- Actions in A: *Putaway* $= \langle H; \neg H \rangle$, *Pick-above-*$x = \langle \neg H, n > 0; H, n{\downarrow} \rangle$, *Put-above-*$x = \langle H; \neg H, n{\uparrow} \rangle$, and *Pick-other* $= \langle \neg H; H \rangle$



QNP Example: Solution

Observation projection for $Q_{clear} = (F, V, I, A, G)$ where

- $F = \{H\}$ and $V = \{n\}$
- $I = \{\overline{H}, n > 0\}$ and $G = \{n = 0\}$
- Actions in A: *Putaway* $= \langle H; \neg H \rangle$, *Pick-above-*$x = \langle \neg H, n > 0; H, n{\downarrow} \rangle$, *Put-above-*$x = \langle H; \neg H, n{\uparrow} \rangle$, and *Pick-other* $= \langle \neg H; H \rangle$



QNP Syntax

QNP is tuple $Q = (F, V, I, A, G)$ where

- F is a finite set of propositions
- V is a finite set of numerical variables
- I is a set of $F{+}V$-literals, where V-literals are $X{=}0$ or $X{>}0$ for X in V
- G is goal condition given by $F{+}V$-literals
- A is set of actions. Each has **precondition** *Pre* ($F{+}V$-literals), **boolean effects** *Eff* (F-literals), and **numerical effects** N (atoms $X{\uparrow}$ or $X{\downarrow}$) with restriction that if $X{\downarrow}$ in N, then $X > 0$ must be in *Pre*

Numerical vars affected only **qualitatively**, and tested for zero

Plan-existence for QNPs **decidable** [Srivastava et al., 2011] whereas it is **undecidable** for numerical planning [Helmert, 2002]

Trajectory Constraints for QNPs

Only **global property** that needs to be accounted for is: for each var X,

> *If X is decremented infinitely often but incremented only a finite number of times, eventually $X = 0$*

Constraint can be expressed in **Linear Temporal Logic (LTL)**, enabling automata-based approaches for solving QNPs [B. et al., 2017]

Theorem

A solution for QNP $Q = (F, V, I, A, G)$ can be obtained in time **doubly exponential** *in $|V| \times |A| + |G|$ and* **exponential** *in $|F| + |V|$*

Automata-based methods need to compute abstraction in explicit form and thus require **exponential space**

First-Order Relational Instances

QNPs also used on STRIPS domains where $\mathcal{Q}$ contains **ground instances** of common schema

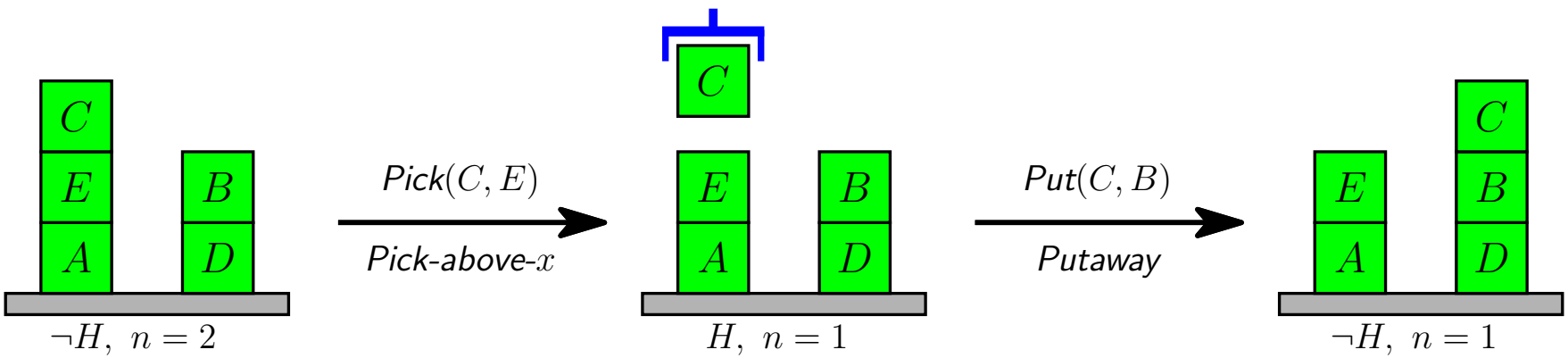
Each task P is associated with **interpretations** ϕ_p and ϕ_X for booleans p and numerical vars X that map embed P-states s into Q-states $\phi(s)$

Abstract actions in Q must **track** value change of variables

Sound Abstraction for Blocksworld

Q_{clear} is **sound abstraction** for Blocksworld instances defined with predicates *on*$(?x, ?y)$, *clear*$(?x)$, *ontable*$(?x)$ and *holding*$(?x)$

Q_{clear} has $F = \{H\}$ and $V = \{n\}$: H captures when gripper is holding a block, and n counts the number of blocks above block A



Pick-above-x **represents** *Pick*(C, E) while *Putaway* **represents** *Put*(C, B)

Sound Abstractions

QNP Q is **sound abstraction** for STRIPS instance P iff

- if s_0 is initial state in P, then $\phi(s_0)$ is initial state in Q
- for each state s in P, if $\phi(s) \models G$ then s is goal in P
- for any reachable state s in P and any abstract action $\bar{a}$ in Q that is **applicable** in $\phi(s)$, there is action a in P that is **represented** by $\bar{a}$ at s

[B. & Geffner, 2018]

Theorem

If Q is sound for P and π solves Q, then the plan π' defined by $\pi'(s) = a$ where a is an action in P that is represented by $\pi(\phi(s)$ at s, **solves** *P. That is, π' solves $\mathcal{Q} = \{P : Q$ is sound abstraction for $P\}$*

Example: Gripper in STRIPS

Problem: Robot with grippers whose goal is to move balls from B to A:

STRIPS schema for Gripper:

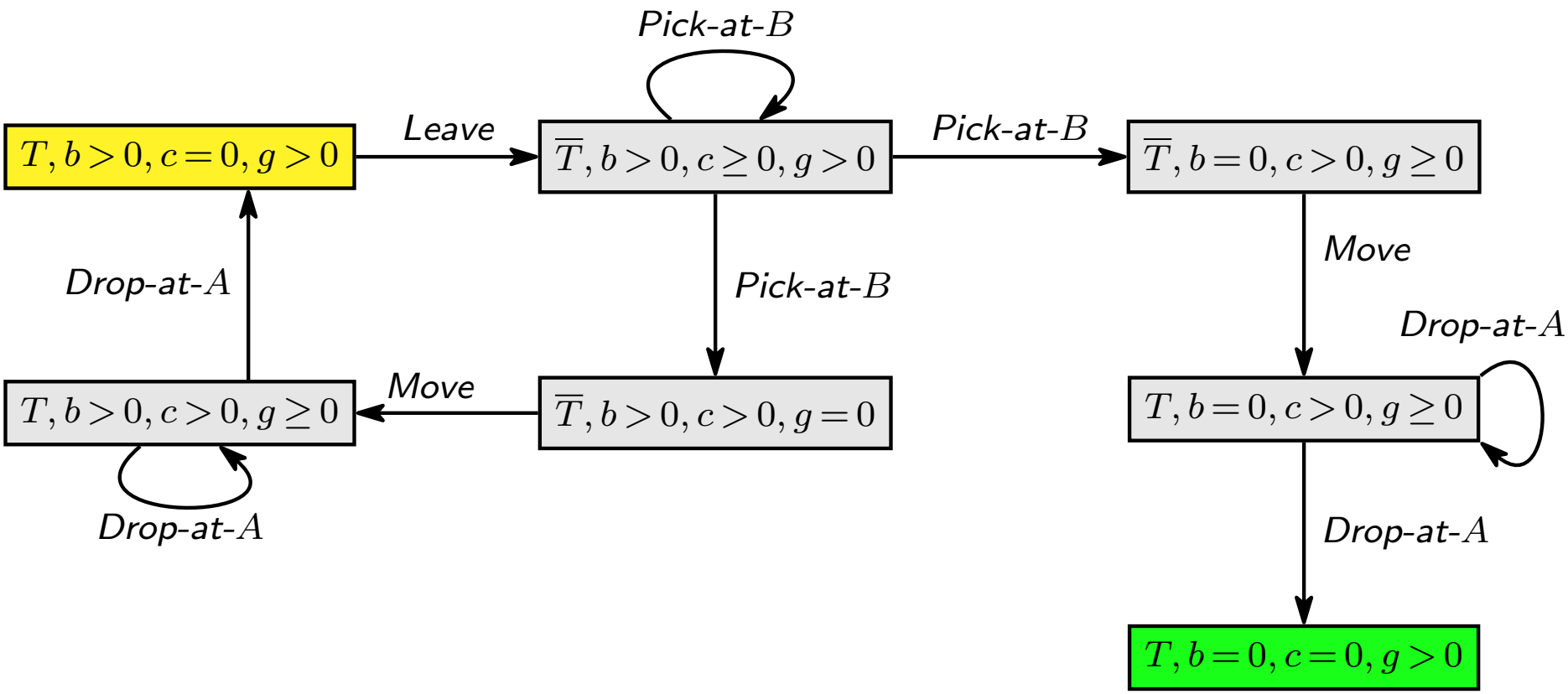
- Atoms: $\mathit{at}(?r)$, $\mathit{ball}(?b, ?r)$, $\mathit{empty}(?g)$, $\mathit{hold}(?b, ?g)$
- Actions:
  - $\mathit{Move}(?s, ?t) = \langle \mathit{at}(?s); \neg\mathit{at}(?s), \mathit{at}(?t) \rangle$
  - $\mathit{Pick}(?b, ?r, ?g) = \langle \mathit{empty}(?g), \mathit{at}(?r), \mathit{ball}(?b, ?r); \neg\mathit{empty}(?g), \neg\mathit{ball}(?b, ?r), \mathit{hold}(?b, ?g) \rangle$
  - $\mathit{Drop}(?b, ?r, ?g) = \langle \mathit{hold}(?b, ?g), \mathit{at}(?r); \neg\mathit{hold}(?b, ?g), \mathit{ball}(?b, ?r), \mathit{empty}(?g) \rangle$
- Initial state is robot at A and all balls at B
- Goal is to have all balls at A

Example: QNP for Gripper

QNP $Q_{gripper} = (F, V, I, A, G)$:

- $F = \{T\}$ where T iff *at*(A)
- $V = \{b, c, g\}$ where b counts balls at B, c balls held, and g free grippers
- $I = \{T, b > 0, c = 0, g > 0\}$ and $G = \{c = 0, b = 0\}$
- Abstract actions are:
  - *Move* $= \langle \neg T; T \rangle$ and *Leave* $= \langle T; \neg T \rangle$
  - *Pick-at-*$B = \langle \neg T, b > 0, g > 0; b\downarrow, c\uparrow, g\downarrow \rangle$
  - *Drop-at-*$B = \langle \neg T, c > 0; b\uparrow, c\downarrow, g\uparrow \rangle$
  - *Drop-at-*$A = \langle T, c > 0; c\downarrow, g\uparrow \rangle$

Example: Solution for Gripper



Solving QNPs with FOND Planners

Reduction from QNP into FOND planning

Reduction is function T : QNPs $\rightarrow$ FONDs such that:

- **Efficient:** given QNP Q returns FOND problem $P = T(Q)$ in polytime
- **Sound:** P has solution implies Q has solution
- **Complete:** Q has solution implies P has solution
- **Effective:** solution for Q can be recoverd in polytime from solution for P

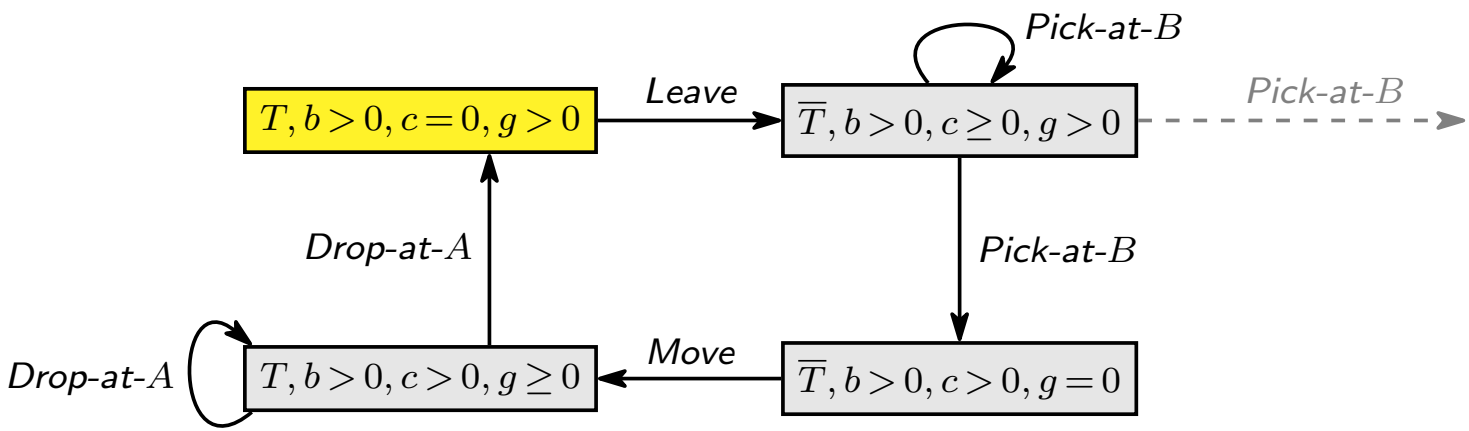
If conditions met, efficient algorithm for QNPs and establish upper bound on complexity

Termination of QNP Plans

QNP solutions are the strong-cyclic solutions that **terminate**

- π terminates if each infinite induced trajectory terminates
- Infinite trajectory denoted by $s_0, s_1, \ldots [s_i, \ldots, s_m]^*$ where $\{s_i, \ldots, s_m\}$ is **set of recurrent states (loop)**
- Such trajectory terminates iff there is variable X that is **decremented but not incremented** in loop

Example: loop terminates because variable b is decremented by *Pick-at-B* but not incremented in loop



Checking Termination with SIEVE

Termination checked with algorithm called SIEVE [Srivastava et al., 2011]

SIEVE works on policy graph $\mathcal{G}(\pi)$: subgraph of projection induced by π

SIEVE (Graph $\mathcal{G}(\pi)$):
repeat

- Compute the strongly connected components (SCC) of $\mathcal{G}(\pi)$
- Choose SCC C and variable X that is **decremented** in C but not **incremented** in C
- Remove edges (s, s') so that s and s' are in C, and $\pi(s)$ decrements X

until $\mathcal{G}(\pi)$ *is acyclic or there is no SCC* C *and variable* X *to choose*

Theorem

π terminates iff SIEVE *reduces $\mathcal{G}(\pi)$ to an acyclic graph. Hence, solution for Q can be obtained in* **exponential space** *by enumerating the strong-cyclic solutions and testing each for termination with* SIEVE

Reduction of QNPs into FOND

Reduction simulates way SIEVE removes edges to reduce SCCs. For this, a **bounded stack** and **bounded counters** are introduced [B. & Geffner, 2020]

Stack used to push/pop numeric vars, while counters used to bound number of operations on stack. The reduction is such that:

- actions that decrement vars must have **at least one such var** in stack
- actions that increment vars must have **none of such variables** in stack

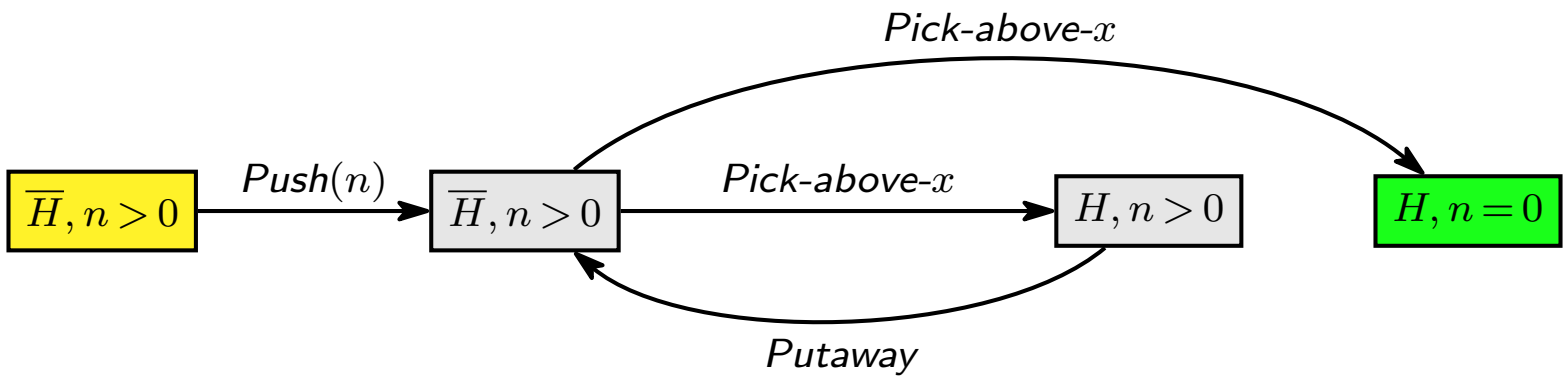
Max stack depth and max counter capacity implemented with polynomial number of propositions and actions

Theorem

QNP Q can be reduced in **polynomial time** *to FOND $P = T(Q)$ such that Q has solution iff P has solution. Moreover, solution for Q can be recovered in polytime from solution for P. Hence, QNPs can be solved in (worst-case)* **exponential time** *since FONDs can be solved in* **exponential time**

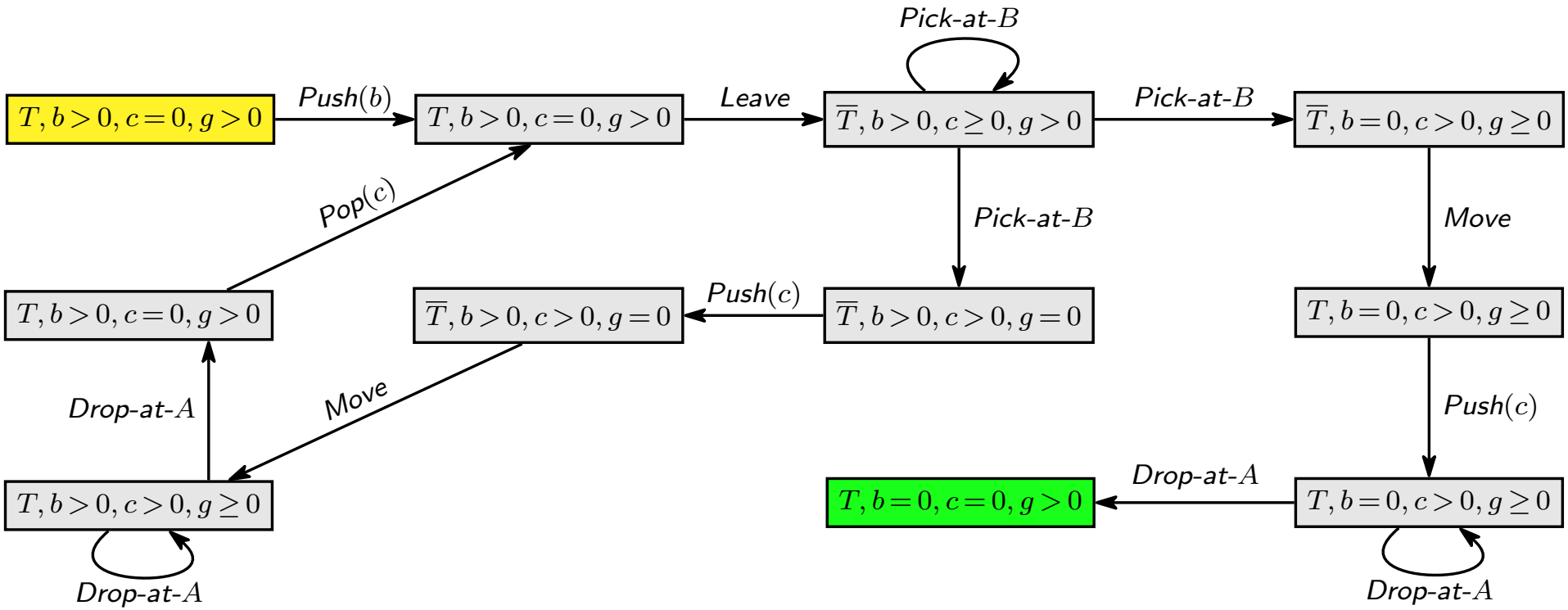
Example: Solving Clear-x

Q_{clear} fed to qnp2fond to get P solved with FOND-SAT [Geffner & Geffner, 2018]



qnp2fond translates Q in less than 0.01 secs. FOND-SAT solves P in 0.08 secs after 3 calls to SAT solver

Example: Solving Gripper



qnp2fond translates Q in less than 0.01 secs. FOND-SAT solves P in 11.25 secs after making 10 calls to SAT solver

Expressiveness and Limitations

Any $\mathcal{Q}$ can be captured with simple QNP Q with only **one variable** d:

– $I = \{d > 0\}$, $G = \{d = 0\}$ and **single action** *Get-closer* $= \langle d > 0; d\downarrow \rangle$

– d is **number of steps to goal**

Interpretation $\phi_d(s)$ performs computation that is **intractable:** compute distance in implicit graph

Makes sense to focus on abstractions whose features are **polynomial-time computable** from representation of states

Outline Recap

1. Non deterministic abstractions for families of classical planning problems
2. Qualitative Numerical Planning (QNP): concrete language for abstractions
3. First-order relational planning instances
4. Solving QNPs via reduction to FOND planning
5. Learning QNP abstractions from traces
6. Wrap Up

Learning QNP Representations

QNPs are expressive and effective!

QNP models can be **learned from samples** [B., Francès & Geffner, 2019]

Main challenge is to learn **concepts** that define features (booleans and numericals); e.g. number of free grippers, distance to target, etc

From pool of concepts that define pool of features, select **minimal subset** that account for transitions in **sampled trajectories**

System learns QNP models that are translated into FOND and solved

QNP Learner

Input: STRIPS domain, sampled trajectories for small instances, and complexity bound N for concepts

Output: QNP model Q that explain trajectories

Method:

- Use atom schemas and fixed **concept grammar** to generate **pool of concepts:** all concepts with complexity $\leq N$
- Interpretation of concept C at state s is **subset of objects** $C(s)$; these define features:
  - **boolean feature** p when $|C(s)| \in \{0, 1\}$ for all states s
  - **numerical feature** $n = |C(s)|$ when $|C(s)| > 1$ for at least some state s
- From **pool of features**, construct SAT theory T that "separates" states and transitions in sample
- Recover QNP model from solution of SAT theory T

SAT Theory for Learning QNPs

Input:

– $\mathcal{S}$ = sample of transitions (s, s') from small instances in $\mathcal{Q}$
– $\mathcal{F}$ = pool of features f together with interpretations $f(s)$ at each s in $\mathcal{S}$

Propositional variables:

– $selected(f)$ for each f in $\mathcal{F}$ to select $\mathcal{F}$-subset
– $D_1(s, t)$ iff selected features distinguish states s and t in $\mathcal{S}$
– $D_2(s, s', t, t')$ iff selected features distinguish (s, s') and (t, t') in $\mathcal{S}$

Formulas:

– $D_1(s, t)$ (for states s and t such that only one is goal)
– $\bigwedge_{t'} D_2(s, s', t, t') \implies D_1(s, t)$ (for each (s, s') and t in $\mathcal{S}$)
– $D_1(s, t) \iff \bigvee_f selected(f)$ (for f's that distinguish s and t)
– $D_2(s, s', t, t') \iff \bigvee_f selected(f)$ (for f's that dist. (s, s') and (t, t'))

Theorem

$T(\mathcal{S}, \mathcal{F})$ is SAT iff there is **sound QNP** *abstraction relative to $\mathcal{S}$ and $\mathcal{F}$*

Learned QNPs: Gripper

Training set: 2 instances with 4 and 5 balls each

Learned features (selected) from $|\mathcal{S}| = 403$ and $|\mathcal{F}| = 130$:

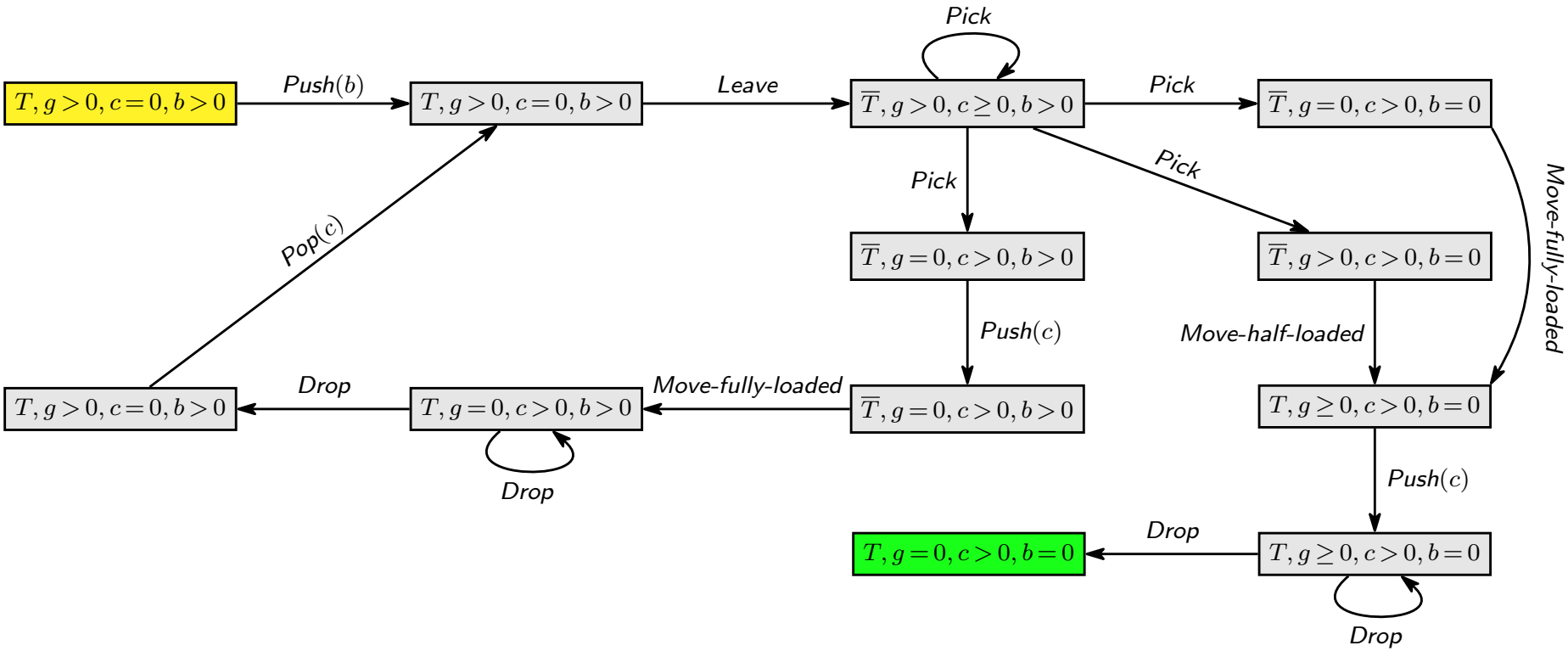
- $T =$ "whether robot is in target room"
- $b =$ "number of balls not in target room"
- $c =$ "number of balls being held by robot"
- $g =$ "number of free grippers (available capacity)"

Learned abstract actions:

- *Drop* $= \langle T, c > 0; c\downarrow, g\uparrow \rangle$
- *Move-fully-loaded* $= \langle \neg T, c > 0, g = 0; T \rangle$
- *Move-half-loaded* $= \langle \neg T, c > 0, g > 0, b = 0; T \rangle$
- *Pick* $= \langle \neg T, b > 0, g > 0; b\downarrow, g\downarrow, c\uparrow \rangle$
- *Leave* $= \langle T, c = 0, g > 0; \neg T \rangle$

Solution works for any number of balls and grippers!

Example: Solution for (Learned) Gripper



Model Q learned and translated into FOND in less than 1 sec. `FOND-SAT` solves the FOND problem in less than 13 secs after 11 calls to SAT solver

Learned QNPs: Pick Rewards in Grid

Inspired from RL work [Garnelo, Arulkumaran & Shanahan, 2016]

Training set: 2 instances 4×4, 5×5, diff. dist. of blocked cells and rewards

Learned Features (selected) from $|\mathcal{S}| = 568$ and $|\mathcal{F}| = 280$:

– $r =$ "number of remaining rewards"

– $d =$ "minimum distance to closest reward along unblocked path"

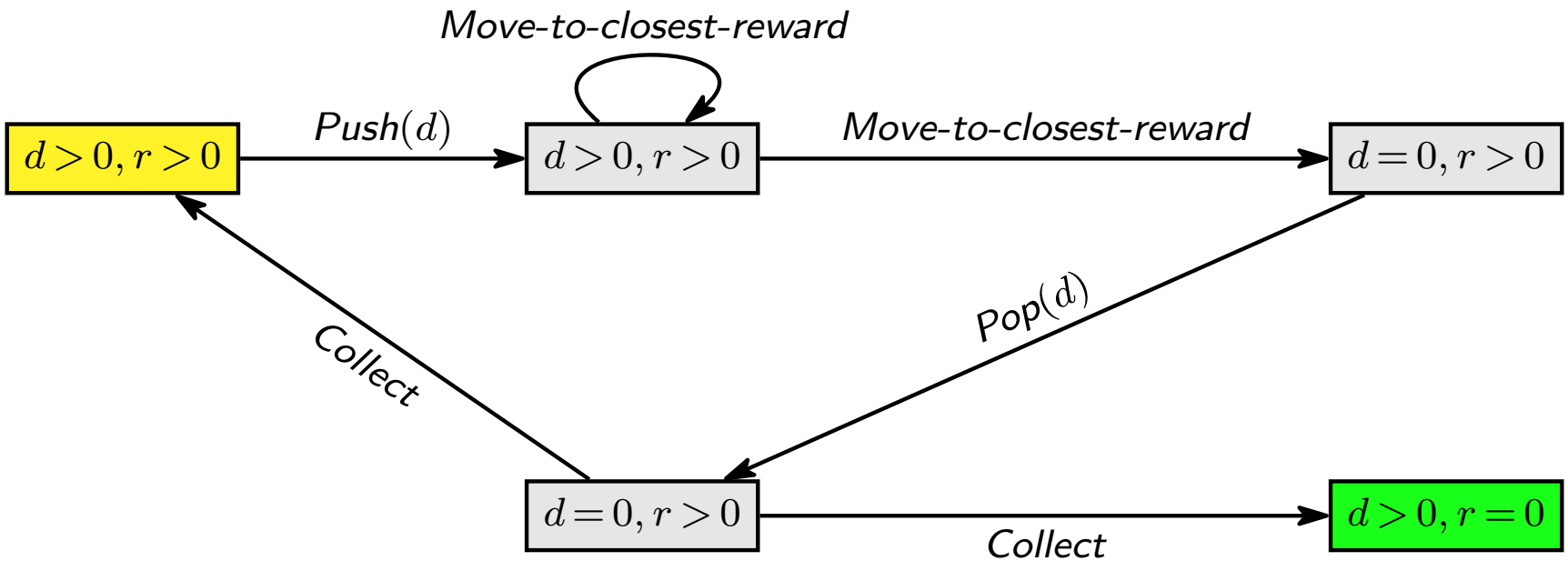
Learned abstract actions:

– *Move-to-closest-reward* $= \langle r > 0, d > 0; d{\downarrow} \rangle$

– *Collect* $= \langle d = 0, r > 0; r{\downarrow}, d{\uparrow} \rangle$

Solution works for any grid dimension, number of rewards, and distribution of blocked cells!

Example: Solution for (Learned) Rewards



Model Q is learned and translated into FOND is less than 1 sec. `FOND-SAT` solves the FOND problem in less than 1 sec after 4 calls to SAT solver

Wrap Up

- Collections of planning problems expressed as non-det abstractions with trajectory constraints
- QNP is suitable language for expressing abstractions and constraints compactly
- QNPs solved with reduction to FOND
- QNP models can be learned

Related Work

- LTL can be used to express more general abstractions and constraints [B. et al, 2017, Aminof et al., 2019; Illanes & McIlraith, 2019, Camacho et al, 2018]
- Model-based learning of general policies from examples [Khardon, 1999; Martín & Geffner, 2004; Fern et al., 2004; Jiménez et al., 2019]
- Deep RL for general policies [Toyer et al., 2018; Issakkimuthu et al., 2018; Bueno et al., 2019; Garg et al., 2020]
- Learning general policies from images without PDDL/STRIPS first-order representations [Ross et al, 2011; Hussein et al., 2017] and DRL

Current Work

Learn QNP models directly from image traces provided by a teacher:

- Use SAT to learn QNP model + solution that **imitates** teacher on traces
- Model has **uninterpreted** variables, but SAT solution gives interpretations for images in traces
- Use interpretations provided by SAT to learn **interpretation functions:**
  - Classifier from state images into valuations for variables
  - Classifier from image transitions into qualitative effects
- Solution + classifiers sufficient for applying policy on new instances

SAT task similar and inspired by work on learning STRIPS representations from non-symbolic traces (ECAI 2020)

Bibliography (1 of 3)

- Aminof, De Giacomo, Murano & Rubin. 2019. Planning under LTL environment specifications. In Proc. ICAPS, pp. 31–39.
- Belle & Levesque. 2016. Foundations for generalized planning in unbounded stochastic domains. In KR, pp. 380–389.
- Bonet, De Giacomo, Geffner & Rubin. 2017. Generalized planning: Non-deterministic abstractions and trajectory constraints. In Proc. 26th Int. Joint Conf. on Artificial Intelligence (IJCAI), pp. 873–879.
- Bonet, Francès & Geffner. 2019. Learning features and abstract actions for computing generalized plans. In Proc. 33rd AAAI Conf. on Artificial Intelligence (AAAI), pp. 2703–2710.
- Bonet, Fuentetaja, E-Martín & Borrajo. 2017. Guarantees for sound abstractions for generalized planning. In Proc. 28th Int. Joint Conf. on Artificial Intelligence (IJCAI), pp. 1566–1573.
- Bonet & Geffner. 2015. Policies that generalize: Solving many planning problems with the same policy. In IJCAI, pp. 2798–2804.
- Bonet & Geffner. 2018. Features, projections, and representation change for generalized planning. In Proc. of the 27th Int. Joint Conf. on Artificial Intelligence (IJCAI), pp. 4667–4673. AAAI Press.
- Bonet & Geffner. 2020. Qualitative Numerical Planning: Reductions and Complexity. JAIR. Forthcoming.
- Bueno, de Barros, Mauá & Sanner. 2019. Deep reactive policies for planning in stochastic nonlinear domains. In AAAI, volume 33, 7530–7537.

Bibliography (2 of 3)

- Camacho, Muise, Baier & McIlraith. 2018. LTL Realizability via Safety and Reachability Games. In IJCAI, 4683–4691.
- Chevalier-Boisvert, Bahdanau, Lahlou, Willems, Saharia, Nguyen & Bengio. 2019. BabyAI: A platform to study the sample efficiency of grounded language learning. In ICLR.
- Espeholt, Soyer, Munos et al. 2018. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. arXiv preprint arXiv:1802.01561.
- François-Lavet, Bengio, Precup & Pineau. 2019. Combined reinforcement learning via abstract representations. In Proc. AAAI, volume 33, 3582–3589.
- Cimatti, Pistore, Roveri & Traverso. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. Artificial Intelligence, 147(1-2), 35–84.
- Garg, Bajpai & Mausam. 2020. Symbolic Network: Generalized Neural Policies for Relational MDPs. arXiv preprint arXiv:2002.07375.
- Garnelo, Arulkumaran & Shanahan. 2016. Towards deep symbolic reinforcement learning. arXiv preprint arXiv:1609.05518.
- Geffner & Geffner. 2018. Compact policies for fully observable non-deterministic planning as sat. In Proc. ICAPS.
- Groshev, Goldstein, Tamar, Srivastava & Abbeel. 2018. Learning generalized reactive policies using deep neural networks. In Proc. ICAPS.
- Helmert. 2002. Decidability and undecidability results for planning with numerical state variables. In Proc. AIPS, pp. 44–53.

Bibliography (3 of 3)

- Hu & De Giacomo. 2011. Generalized planning: Synthesizing plans that work for multiple environments. In IJCAI, pp. 918–923.
- Hussein, Gaber, Elyan & Jayne. 2017. Imitation learning: A survey of learning methods. ACM Computing Surveys 50(2):1–35.
- Illanes & McIlraith. 2019. Generalized planning via abstraction: arbitrary numbers of objects. In Proc. AAAI.
- Issakkimuthu, Fern & Tadepalli. 2018. Training deep reactive policies for probabilistic planning problems. In ICAPS.
- Jiménez, Segovia-Aguas & Jonsson. 2019. A review of generalized planning. The Knowledge Engineering Review, 34.
- Khardon. 1999. Learning action strategies for planning domains. Artificial Intelligence 113(1-2):125–148.
- Levesque. 2005. Planning with loops. In IJCAI, pp. 509–515.
- Martín & Geffner. 2004. Learning generalized policies from planning examples using concept languages. Applied Intelligence 20(1):9–19.
- Ross, S.; Gordon, G.; and Bagnell, D. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In Proc. AI and Statistics, 627–635.
- Srivastava, Zilberstein, Immerman & Geffner. 2011. Qualitative numeric planning. In AAAI.