Learning Features and Abstract Actions for Computing Generalized Plans

Blai Bonet
Universidad Simón Bolívar, Venezuela

Guillem Francès
University of Basel, Switzerland

Hector Geffner
ICREA & Universitat Pompeu Fabra, Spain

Planning and Generalized Planning

- **Planning** is about solving **single planning instances**
  - ▷ E.g., find **plan** to achieve $on(A, B)$ for **particular configuration of blocks**

- **Generalized planning** is about solving **multiple planning instances** at once.

  E.g., find **general strategy** for

1. go to target location (x^*, y^*) in empty square grid of **any** size
2. pick objects spread in 2D grid, **any** number, size, locations
3. achieve goal $on(x, y)$ in Blocks, **any** number of blocks and configuration
4. achieve **any goal** in Blocks, **any** number of blocks, any configuration, . . .

Srivastava et al, 2008; Bonet et al, 2009; Hu and De Giacomo 2011, . . .

Generalized Planning: Motivation

- **Broaden scope of planners:** General strategies for playing Atari games?
- **Insight into representations:** What representations adequate and why?
- **Connections with (deep) learning:** How to learn general features and plans?

Outline of the Talk

- **Generalized planning: basic formulation**
- **Extended formulation. abstract actions**
- **Learning** features and abstract actions
- Wrap Up, Future

Generalized Planning: Basic Formulation

- **Generalized** $\mathcal{Q}$ is set of planning instances P sharing **actions** and **features**
- **Features** f represent **state functions** $\phi_f(s)$ over finite domains (observations)
- **Policy** for $\mathcal{Q}$ is mapping π from **feature valuations** into **actions**
- **Solutions:** π solves **general** $\mathcal{Q}$ iff π solves **each** $P \in \mathcal{Q}$

Example

- **Task** $\mathcal{Q}_{hall}$: Clean cells in $1 \times n$ hall, starting from left, **any** n and dirt
- **Features** d, e: if current cell is dirty, if current cell is last
- **Actions** $move$, $clean$: move right, clean current cell
- **Solution:** Policy "If d, $clean$", "If $\neg d$ and $\neg e$, $move$"

Generalized Planning: Extended Formulation, BG 2018

- **Generalized** $\mathcal{Q}$ is set of planning instances P sharing set of **features** F
- **Features** can be **boolean** p or **numerical** n with functions $\phi_p(s)$ and $\phi_n(s)$
- **Boolean feature valuation** assigns truth values to atoms $X_p = true$ and $X_n = 0$
- **Sound abstract actions on feature variables that track value of features**

- **Policy** π for $\mathcal{Q}$ maps **boolean feature valuations** into **abstract actions**
- **Solutions:** π solves **general** $\mathcal{Q}$ iff π solves **each** $P \in \mathcal{Q}$

Abstract Actions: Language

- Features F will refer to **state functions** $\phi_f(s)$ in instances P but to **state (feature) variables** in **abstraction**

- **Abstract action** $\bar{a} = Pre \mapsto Eff$ defined over **feature variables**:
  - ▷ **Boolean preconditions and effects:** p, $\neg p$
  - ▷ **Numerical preconditions:** $n = 0$, $n > 0$
  - ▷ **Numerical effects:** $n\!\uparrow$, $n\!\downarrow$ (inc's and dec's by **unspecified amounts**)

- Language of **qualitative numerical problems (QNPs)**, Srivastava et al, 2011
  - ▷ Sufficiently **expressive** for abstraction
  - ▷ Compiles into **fully observable non-deterministic (FOND)** planning:
  - ▷ ($n = 0$ become boolean var, $n > 0$ its negation, effect $n\!\uparrow$ becomes $n > 0$, and $n\!\downarrow$, $n = 0 \,|\, n > 0$; non-det effects $n = 0 \,|\, n > 0$, however, **conditionally fair**)

Abstract Actions: Soundness and Completeness

They enable us to reason about all instances P in parallel, in terms of **abstract actions that operate at the level of features**

Def: Action b and abstract $\bar{a} = Pre \mapsto \mathit{Eff}$ **have same effects over** F in state s, if both applicable in s with **same effects on the features** p **and** n **in** F**:**

1. $p \in \mathit{Eff}$ iff $\phi_p(s')$ true and $\phi_p(s)$ false; $s' = f(b, s)$
2. $\neg p \in \mathit{Eff}$ iff $\phi_p(s')$ false and $\phi_p(s)$ true; $s' = f(b, s)$
3. $n\!\uparrow\, \in \mathit{Eff}$ iff $\phi_n(s') > \phi_p(s)$; $s' = f(b, s)$
4. $n\!\uparrow\, \in \mathit{Eff}$ iff $\phi_n(s') > \phi_p(s)$; $s' = f(b, s)$

Def: Abstract actions A_F **sound** in $\mathcal{Q}$ iff for any s over instance P of $\mathcal{Q}$, if $\bar{a}$ in A_F is applicable in s, there is action b in P with the same effects as $\bar{a}$ in s.

Def: Abstract actions A_F **complete** in $\mathcal{Q}$ iff for any s over instance P of $\mathcal{Q}$, if $\bar{a}$ in A_F is applicable in s, there is action b in P with the same effects as $\bar{a}$ in s.

Computation: Solving Generalized Problem $\mathcal{Q}$

1. Define **abstraction** $Q_F = \langle V_F, I_F, G_F, A_F \rangle$ from features F and **sound** A_F. Initial and goals I_F and G_F to match $\mathcal{Q}$.

2. Abstraction Q_F converted into **FOND** Q'_F by replacing $n \in N$ by symbol $n = 0$, and effects $n\!\uparrow$ and $n\!\downarrow$, by $n > 0$ and $n > 0 \,|\, n = 0$ resp.

3. Amend Q'_F into FOND Q^+_F so that solutions assume **conditional fairness** (infty decrements of n eventually yield $n = 0$, if increments finite).

Theorem (BG 2018): Solutions of FOND Q^+_F computed by FOND planners off-the-shelf are solutions to all instances P in $\mathcal{Q}$.

Example: $\mathcal{Q}_{on(x,y)}$

- **Features** $F = \{n(x), n(y), X, H, on(x,y)\}$; $n(x)$ is # blocks above x
- **Abstract Actions** A_F; E abbreviates $\neg X \wedge \neg H$
  - ▷ Pick-x : $E, n(x) = 0 \;\mapsto\; X$,
  - ▷ Pick-above-x : $E, n(x) > 0 \;\mapsto\; H, n(x){\downarrow}$,
  - ▷ Pick-above-y : $E, n(y) > 0 \;\mapsto\; H, n(y){\downarrow}$,
  - ▷ Put-x-on-y : $X, n(y) = 0 \;\mapsto\; \neg X, on(x,y), n(y){\uparrow}$,
  - ▷ Put-other-aside : $H \;\mapsto\; \neg H$.
- **Abstraction** $Q_F = \langle V_F, I_F, G_F, A_F \rangle$, $I_F = \ldots$ and $G_F = \{on(x,y)\}$
- **FOND** $Q'_F = \langle V'_F, I'_F, G'_F, A'_F \rangle$ with booleans $n(x) = 0$ and $n(y) = 0$ only
- **FOND planner** yields policy π that achieves $on(x,y)$ in 70msecs:
  - ▷ **If** $E, n(x) > 0, n(y) > 0$ **do** Pick-above-x,
  - ▷ **If** $H, \neg X, n(x) > 0, n(y) > 0$ **do** Put-other-aside,
  - ▷ **If** $H, \neg X, n(x) = 0, n(y) > 0$ **do** Put-other-aside,
  - ▷ **If** $E, n(x) = 0, n(y) > 0$ **do** Pick-above-y,
  - ▷ **If** $H, \neg X, n(x) = 0, n(y) = 0$ **do** Put-other-aside,
  - ▷ **If** $E, n(x) = 0, n(y) = 0$ **do** Pick-above-x,
  - ▷ **If** $X, \neg H, n(x) = 0, n(y) = 0$ **do** Put-$x{-}on{-}y$.

Learning Features and Abstract Actions From Samples

Sample $\mathcal{S}$: Finite set of **state transitions** (s, b, s') drawn from instances P in $\mathcal{Q}$ such that states s appearing **first** in transitions, **fully expanded**

Def: Abstract actions A_F **sound** relative to sample $\mathcal{S}$ of $\mathcal{Q}$, iff for any $\bar{a}$ in A_F applicable in $s \in \mathcal{S}$, there is a transition (s, b, s') in $\mathcal{S}$ such that $\bar{a}$ and b have **same effects over features** in s.

Def: Abstract actions A_F **complete** relative to sample $\mathcal{S}$ of $\mathcal{Q}$, iff for any transition (s, b, s') in $\mathcal{S}$, there is $\bar{a}$ in A_F applicable in $s \in \mathcal{S}$, such that b and $\bar{a}$ **have same effects** over features in s.

For sufficiently large sample, approx and exact soundness and completeness converge

Learning F and A_F with SAT: $T(\mathcal{S}, \mathcal{F})$

Variables:

- $selected(f)$ for each $f \in \mathcal{F}$, true iff $f \in F$, $F \subseteq \mathcal{F}$
- $D_1(s,t)$ true iff selected features distinguish s from t; p or $n = 0$ true in one
- $D_2(s,s',t,t')$ true iff selected features f distinguish transitions (s,s'), (t,t')

Formulas:

- $D_1(s,t) \;\Leftrightarrow\; \bigvee_f selected(f)$
- $D_2(s,s',t,t') \;\Leftrightarrow\; \bigvee_f selected(f)$
- $\neg D_1(s,t) \;\Rightarrow\; \bigvee_{t'} \neg D_2(s,s',t,t')$
- $D_1(s,t)$, when one of s and t is a goal state

Theorem: $T(\mathcal{S}, \mathcal{F})$ is SAT iff $\exists$ set of features $F \subseteq \mathcal{F}$ and abstract actions A_F over F such that A_F is **sound and complete** relative to $\mathcal{S}$.

Learning F and A_F via SAT: $T_G(\mathcal{S}, \mathcal{F})$

- Similar result for **smaller theory** $T_G(\mathcal{S}, \mathcal{F})$ that marks some state transition (s, s') as **goal-relevant** with a **planner** on sampled instances P

- Theory $T_G(\mathcal{S}, \mathcal{F})$ ensures **soundness** over $\mathcal{S}$ but **completeness** over marked transitions in $\mathcal{S}$.

- $T_G(\mathcal{S}, \mathcal{F})$ is like $T(\mathcal{S}, \mathcal{F})$ but transitions (s, s') in $D_2(s, s', t, t')$ range over goal relevant pairs only.

Feature Pool

- Pool of candidate features $\mathcal{F}$ in $T(\mathcal{S}, \mathcal{F})$ def'ed from **primitive predicates** in $\mathcal{Q}$
- **Boolean and numerical features** b_r and n_c defined from **unary predicates** r:
  - ▷ $\phi_{b_r}(s) = (|r^s| > 0)$ and $\phi_{n_r}(s) = |r^s|$ if $r^s = \{c \,|\, r(c)$ is true in $s\}$
- **New unary predicates** r generated from **description logic grammar**
  - ▷ $C \;\leftarrow\; C_p, C_u, C_x$, primitive, universal, parameter
  - ▷ $C \;\leftarrow\; \neg C, C \sqcap C'$, negation, conjunction
  - ▷ $C \;\leftarrow\; \exists R.C, \forall R.C$, existential and universal roles
  - ▷ $R \leftarrow R_p$, R_p^{-1}, R_p^*, $[R_p^{-1}]^*$: primitive, inverse, closure
- $dist(C_1, R:C, C_2)$ represents min n s.t. $C_1^s(x_1)$, $C_2^s(x_n)$, and $(R:C)^s(x_i, x_{i+1})$
- Max SAT solver **minimizes** $\sum_{f:selected(f)} cost(f)$, given by structure of f

Computational Model Updated: Learn then Plan

For solving generalized problem $\mathcal{Q}$:

1. sample set of transition $\mathcal{S}$ from instances P
2. compute pool of features $\mathcal{F}$ from primitive predicates, grammar, bounds
3. Max SAT to find assignment of $T(\mathcal{S},\mathcal{F})$ or $T_G(\mathcal{S},\mathcal{F})$ that min $\sum_{f \in F_\sigma} cost(f)$
4. extract features F and abstract actions A_F from assignment
5. define abstraction $Q_F = \langle V_F, I_F, G_F, A_F \rangle$ with I_F and G_F to match $\mathcal{Q}$,
6. reduce Q_F to FOND Q_F^+

Theorem: If abstract actions A_F that are sound relative to the sample $\mathcal{S}$ are sound relative to $\mathcal{Q}$, then the policy π that solves Q_F^+ solves all instances P of $\mathcal{Q}$.

Experimental Results: Problem Data

	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{F}\rvert$	$T(\mathcal{S},\mathcal{F})$ np	$T(\mathcal{S},\mathcal{F})$ nc	$T_G(\mathcal{S},\mathcal{F})$ np	$T_G(\mathcal{S},\mathcal{F})$ nc	t_{SAT}	$\lvert F\rvert$	$\lvert A_F\rvert$	t_{FOND}	$\lvert\pi\rvert$
$\mathcal{Q}_{clear}$	927	322	535K	59.6M	7.7K	767K	0.01	3	2	0.46	5
$\mathcal{Q}_{on}$	420	657	128K	25.8M	18.3K	3.3M	0.02	5	7	7.56	12
$\mathcal{Q}_{grip}$	403	130	93K	4.7M	8.1K	358K	0.01	4	5	171	14
$\mathcal{Q}_{rew}$	568	280	184K	11.9M	15.9K	1.2M	0.01	2	2	1.36	7

n is # of training instances P, $|\mathcal{S}|$ is # of transitions in $\mathcal{S}$, $|\mathcal{F}|$ is size of pool, np and nc are # of vars and clauses in $T(\mathcal{S},\mathcal{F})$ and $T_G(\mathcal{S},\mathcal{F})$, t_{SAT} is time for SAT solver on T_G, $|F|$ and $|A_F|$ are # of selected features and abstract actions, t_{FOND} is time for FOND planner, and $|\pi|$ is size of resulting policy. Times in seconds.

Experimental Results: $\mathcal{Q}_{on}$

- $\mathcal{Q}_{on}$: STRIPS instances with goal $on(x, y)$, x and y not in same tower initially
- **Training:** 3 instances P, 420 state transitions in $\mathcal{S}$, 657 features in $\mathcal{F}$
- **Features learned** E, X and G: empty gripper, holding x, x on y, $n(x)$, $n(y)$
- **Abstract Actions learned:**

  1. pick-ab-$x = E, \neg X, \neg G, n(x) > 0, n(y) > 0 \mapsto \neg E, n(x)\downarrow$,
  2. pick-ab-$y = E, \neg X, \neg G, n(x) = 0, n(y) > 0 \mapsto \neg E, n(y)\downarrow$,
  3. put-aside-1 $= \neg E, \neg X, \neg G, n(x) = 0 \mapsto E$,
  4. put-aside-2 $= \neg E, \neg X, \neg G, n(x) > 0, n(y) > 0 \mapsto E$,
  5. pick-$x = E, \neg X, \neg G, n(x) = 0, n(y) = 0 \mapsto \neg E, X$,
  6. put-x-aside $= \neg E, X, \neg G, n(x) = 0, n(y) > 0 \mapsto E, \neg X$,
  7. put-x-on-$y = \langle \neg E, X, \neg G, n(x) = 0, n(y) = 0; E, \neg X, G, n(y)\uparrow \rangle$.

- **Abstraction** Q_F with $I_F = \ldots$ and and $G_F = \{G\}$
- **FOND** Q_F^+ in 0.01 seconds, solved by FOND planner in 7.56 secs
- Resulting policy solves all instances P in $\mathcal{Q}_{on}$; **any** number and config of blocks

Experimental Results: $\mathcal{Q}_{gripper}$

- $\mathcal{Q}_{gripper}$: STRIPS instances, $at{-}robby(l)$, $at{-}ball(b,l)$, $free(g)$, $carry(b,g)$
- **Features learned** from 3 instances P, 403 transitions in $\mathcal{S}$, 130 features $\mathcal{F}$

  1. $X : at\_robby \sqcap C_x$ (whether robby is in target room),
  2. $B : |\exists\, at.\neg C_x|$ (number of balls not in target room),
  3. $C : |\exists\, carry.C_u|$ (number of balls carried),
  4. $G : |free|$ (number of empty grippers).

- **Abstract Actions learned**

  1. drop-ball-at-$x = C > 0, X \;\mapsto\; C{\downarrow}, G{\uparrow}$,
  2. move-to-x-half-loaded$=\neg X, B = 0, C > 0, G{>}0 \;\mapsto\; X$,
  3. move-to-x-fully-loaded $= \neg X, C > 0, G = 0 \;\mapsto\; X$,
  4. pick-ball-not-in-$x = \neg X, B > 0, G > 0 \;\mapsto\; B{\downarrow}, G{\downarrow}, C{\uparrow}$,
  5. leave-$x = X, C = 0, G > 0 \;\mapsto\; \neg X$

- **Abstraction** Q_F with $I_F = \ldots$, $G_F = \{B = 0\}$,
- **FOND** Q_F^+ obtained in 0.01 secs and solved by FOND planner in 171.92 secs
- Resulting policy solves $\mathcal{Q}_{gripper}$; **any** number of grippers and balls.

Experimental Results: $\mathcal{Q}_{reward}$

- $\mathcal{Q}_{reward}$: pick rewards spread on grid with blocked cells (from *Towards Deep Symbolic RL*, M. Garnelo, K. Arulkumaran, M. Shanahan, 2016)
- STRIPS instances with predicates $reward(\cdot)$, $at(\cdot)$, $blocked(\cdot)$, $adj(\cdot,\cdot)$
- **Training:** 2 instances 4×4, 5×5, diff distributions of blocked cells and rewards
- **Learned features:**
  1. $R : |reward|$ (number of remaining rewards),
  2. $D : dist(at, adjacent{:}\neg blocked, reward)$
- **Learned abstract actions:**
  1. collect-reward $= D = 0, R > 0 \mapsto R{\downarrow}, D{\uparrow}$,
  2. move-to-closest-reward $= R > 0, D > 0 \mapsto D{\downarrow}$
- **Abstract** Q_F with $I_F = \{R > 0, D > 0\}$ and $G_F = \{R = 0, D > 0\}$
- **Policy** that solves $\mathcal{Q}_{reward}$ obtained from Q_F^+ in 1.36 secs.

Wrap Up: Limitations

- **Computational bottleneck** in size of theories used to derive features and actions
  - ▷ Used theories T_G with marked transitions, not T
  - ▷ While **optimal** Max SAT solvers effective, need to try **suboptimal** solvers
- **Expressive bottleneck: pool of features** from general grammar?
  - ▷ Domains with **bounded width** (Lipovetzky and G., 2012) admit **compact policies** with **poly-time features** ($\bar{a} : n^* > 0 \mapsto n^*\downarrow$)
  - ▷ For arbitrary goals, grammar seems ok, add **goal predicates** (Martin and G., 2000)

Summary and Future

- Scheme for computing **general plans** that mixes **learning** and **planning**:
  - ▷ **Learner** infers abstraction by enforcing **soundness** and **completeness**
  - ▷ **Planner** uses abstraction, transformed, to compute the general plans

- **Number of samples** small as learner identifies features for the planner to track

- Unlike purely learning approaches, features and policies are **transparent**, and **scope and correctness** of plans can be assessed

- Relation to **dimensionality reduction** and **embeddings** in ML/Deep Learning
  - ▷ abstraction maps states into bounded features, preserving essential properties

- **Challenge:** Learn **embeddings** that yield **sound** and **complete** abstractions