Progress and Challenges in Planning

Hector Geffner
ICREA & Universitat Pompeu Fabra
Barcelona, Spain

PFIA-JFPDA Rennes, 7/2015

Joint work with many people: Blai Bonet, Hector Palacios,
Nir Lipovetzky, Miquel Ramirez, Filippos Kominis, Tomas Geffner, . . .

Planning and Autonomous Behavior

Key problem in **autonomous behavior** is **control:** what to do next. Three approaches to this problem:

- **Programming-based:** Specify control by hand
- **Learning-based:** Learn control from experience
- **Model-based:** Specify problem by hand, derive control automatically

Approaches not disjoint; successes and limitations in each . . .

Planning is the **model-based approach** to autonomous behavior; **model** captures **predictions**: what actions do in the world, and what sensors tell us about the world

Wumpus World PEAS description

Performance measure

gold +1000, death -1000

-1 per step, -10 for using the arrow

Environment

Squares adjacent to wumpus are smelly

Squares adjacent to pit are breezy

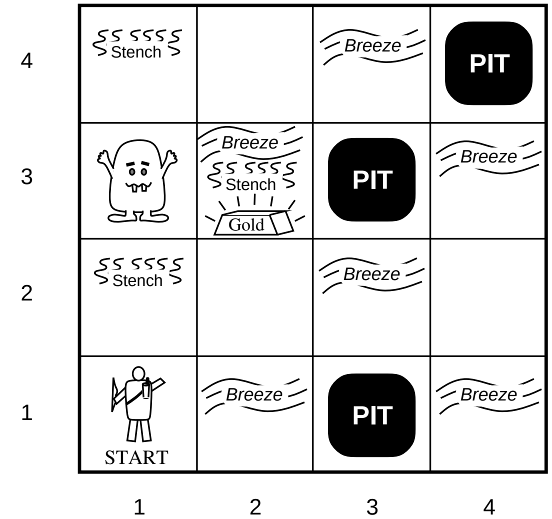
Glitter iff gold is in the same square

Shooting kills wumpus if you are facing it

Shooting uses up the only arrow

Grabbing picks up gold if in same square

Releasing drops the gold in same square



Actuators Left turn, Right turn,

Forward, Grab, Release, Shoot

Sensors Breeze, Glitter, Smell

State Model for (Classical) AI Planning

- finite and discrete state space S
- a **known initial state** $s_0 \in S$
- a set $S_G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a **deterministic state transition function** $s' = f(a, s)$ for $a \in A(s)$
- action costs $c(a, s) > 0$

A **solution** is a sequence of applicable actions that maps s_0 into S_G

It is **optimal** if it minimizes sum of action costs (e.g., $\#$ of steps)

The resulting controller is **open-loop** (no feedback)

Uncertainty but No Feedback: Conformant Planning

- finite and discrete state space S
- a **set of possible initial state** $S_0 \in S$
- a set $S_G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a **non-deterministic** transition function $F(a, s) \subseteq S$ for $a \in A(s)$
- action costs $c(a, s)$

Uncertainty but no sensing; hence controller still **open-loop**

A **solution** is an **action sequence** that achieves the goal in spite of the uncertainty; i.e. for **any possible initial state** and **any possible transition**

Planning with Sensing and POMDPs

- finite and discrete state space S
- a **set of possible initial state** $S_0 \in S$
- a set $S_G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a **non-deterministic** transition function $F(a, s) \subseteq S$ for $a \in A(s)$
- action costs $c(a, s)$

- a **sensor model** $O(a, s)$ mapping actions and states into observation tokens o

Solutions can be expressed in many forms; e.g., **policies** mapping belief states into actions, contingent **trees**, finite-state **controllers**, etc.

Probabilistic version known as **POMDP**: Partially Obs. Markov Decision Process

Models, Languages, Control, Scalability

- A **planner** is a **solver over a class of models;** it takes a model description, and computes the corresponding control

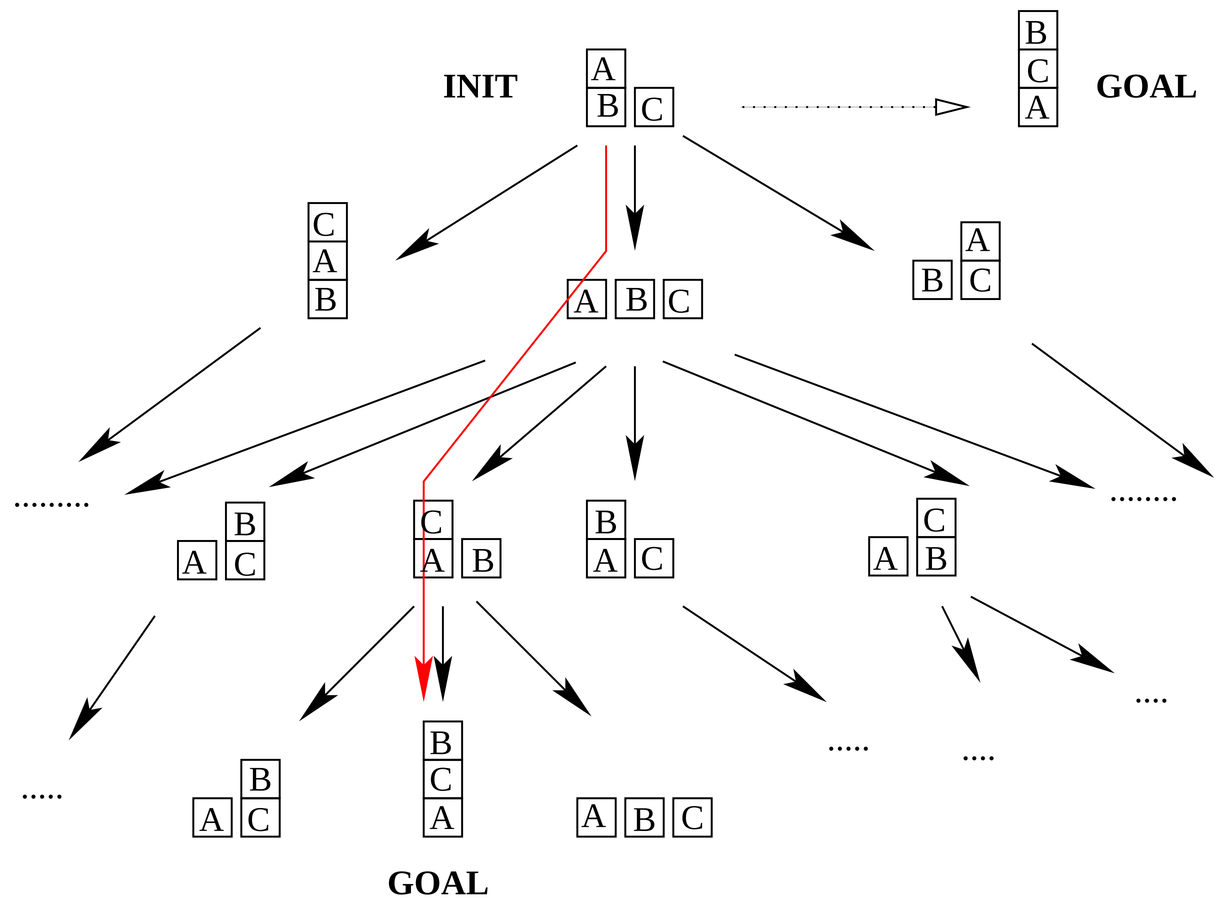
- Many dimensions and models: **uncertainty**, **feedback**, **costs**, . . .

- Models described in compact form by means of **planning languages**

- Different **types** of control:
  - ▷ **open-loop** vs. **closed-loop** (feedback used)
  - ▷ **off-line** vs. **on-line** (full policies vs. lookahead)

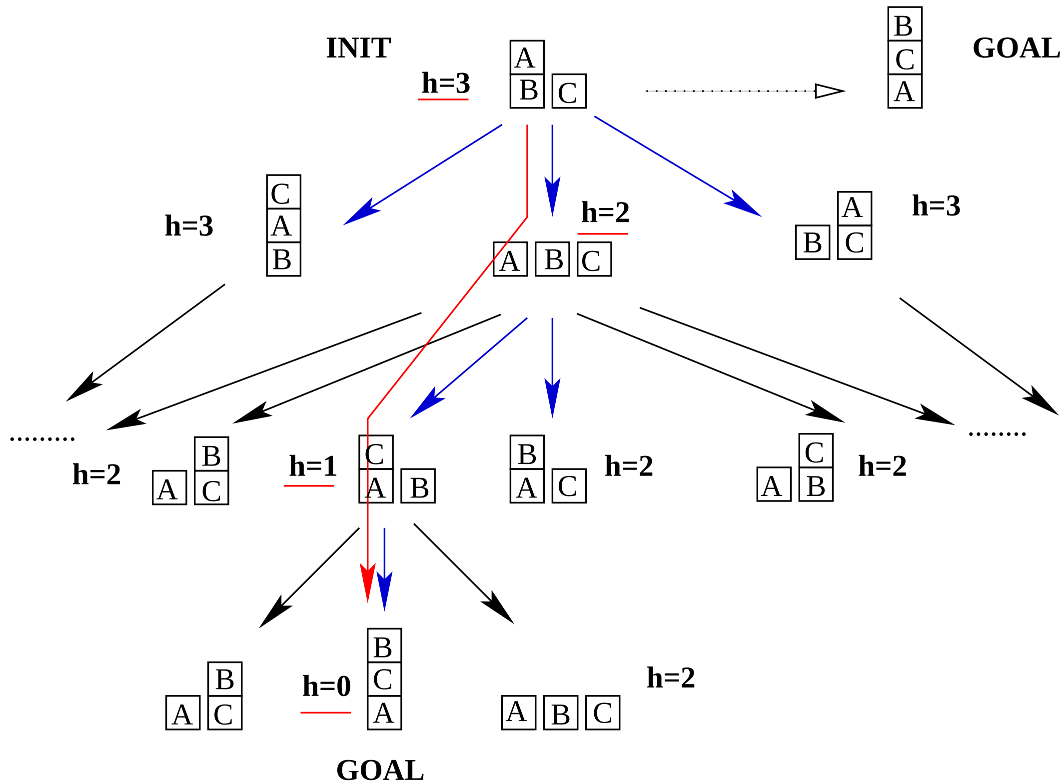
- All models **intractable**; key challenge is **scalability**
  - ▷ how not to be defeated by **problem size**
  - ▷ need to exploit the **structure** of problems

Combinatorial Explosion: Example



- **Classical** problem: move blocks to transform **Init** into **Goal**
- Problem becomes **path finding** in **directed graph** associated with $\mathcal{S}(P)$
- Difficulty is that graph size is **exponential** in number of blocks
- Problem simple for specialized Block Solver but difficult for General Solver

Dealing with the Combinatorial Explosion: Heuristics



- **Heuristic values** $h(s)$ estimate "cost" from s to goal; provide sense of direction
- They are **derived automatically** from problem representation
- Plans can be found then with **heuristic search**

Status AI Planning

- **Classical planners work reasonably well**
  - ▷ *Large problems solved very fast* (non-optimally)
  - ▷ *Different types of* **inference**: *heuristics, landmarks, helpful actions*
  - ▷ *Specialized* **SAT** *approaches work well too (Rintanen)*

- **Model simple but useful**
  - ▷ *Operators not primitive; can be policies themselves*
  - ▷ *Fast closed-loop replanning able to cope with uncertainty sometimes*

- **Beyond Classical Planning:** incomplete information, uncertainty, . . .
  - ▷ **Top-down** *approach: general* **native solvers** *for MDPs, POMDPs, etc.*
  - ▷ **Bottom-up** *approach:* **transformations** *and use of classical planners*

Next: Three Simple, Crisp Ideas that Appear to be Practical

- Classical planning is **PSPACE** but problems appear to be **easy**. **Why?**
  - ▷ **Width-based search** for classical planning
  - ▷ Results in **Atari** and **General Video Games** (ALE, GVG-AI)
    (Lipovetzky and G. ECAI-2012, Lipovetzky et al IJCAI-2015)

- How to **scale up** when planning with **partial observability**?
  - ▷ **Automatic transformations** and use of **classical planners**
  - ▷ Results in domains like **Wumpus** and **Minesweeper**
    (Bonet and G. IJCAI-2011, AAAI-2014)

- Planning with **nested beliefs** in presence of **multiple agents**
  - ▷ **Formulation** that can be compiled into **classical planning**
  - ▷ **Language**, **semantics**, and **computation**
    (Kominis and G. ICAPS-2015)

IW: A Stupid but Powerful Blind-Search Algorithm?

Define the **novelty** of a newly generated state s in the search as the **size of the smallest tuple of atoms** t that is **true** in s and **false** in all previously generated states s'.

E.g., if s makes some **atom** true for the first time, then novelty of s is 1, else if s makes some **pair of atoms** true for the first time, novelty of s is 2, etc.

Iterative Width (IW):

- **IW**(i) is a **breadth-first** search that **prunes** newly generated states s with $novelty(s) > i$.
- **IW**(i) runs is **exponential in** i, **not** in number of variables as **normal BrFS**
- **IW** is **sequence of calls IW**(i) for $i = 1, 2, \ldots$ over problem P until problem solved or i exceeds number of variables in problem

How well does IW do? Planning with atomic goals

#	Domain	I	$\mathbf{IW}(1)$	$\mathbf{IW}(2)$	Neither
1.	8puzzle	400	55%	45%	0%
2.	Barman	232	9%	0%	91%
3.	Blocks World	598	26%	74%	0%
4.	Cybersecure	86	65%	0%	35%
...	...	...	...	...	
22.	Pegsol	964	92%	8%	0%
23.	Pipes-NonTan	259	44%	56%	0%
24.	Pipes-Tan	369	59%	37%	3%
25.	PSRsmall	316	92%	0%	8%
26.	Rovers	488	47%	53%	0%
27.	Satellite	308	11%	89%	0%
28.	Scanalyzer	624	100%	0%	0%
...	...	...	...	...	
33.	Transport	330	0%	100%	0%
34.	Trucks	345	0%	100%	0%
35.	Visitall	21859	100%	0%	0%
36.	Woodworking	1659	100%	0%	0%
37.	Zeno	219	21%	79%	0%
	Total/Avgs	37921	37.0%	51.3%	11.7%

# Instances	**IW**	ID	**BrFS**	*GBFS* + h_{add}
37921	34627	9010	8762	34849

Top: Instances solved by IW(1) and IW(2). **Bottom:** Comparison with ID, BrFS, and GBFS with h_{add} (Lipovetzky & G. 2012)

Sequential IW: Using IW Sequentially to Solve Joint Goals

SIW runs **IW** sequentially for achieving **one (more) goal at a time** (hill-climbing)

		Serialized **IW** (SIW)				GBFS + h_{add}		
Domain	I	S	Q	T	M/Aw_e	S	Q	T
8puzzle	50	50	42.34	0.64	4/1.75	50	55.94	0.07
Blocks World	50	50	48.32	5.05	3/1.22	50	122.96	3.50
Depots	22	21	34.55	22.32	3/1.74	11	104.55	121.24
Driver	20	16	28.21	2.76	3/1.31	14	26.86	0.30
Elevators	30	27	55.00	13.90	2/2.00	16	101.50	210.50
Freecell	20	19	47.50	7.53	2/1.62	17	62.88	68.25
Grid	5	5	36.00	22.66	3/2.12	3	195.67	320.65
OpenStacksIPC6	30	26	29.43	108.27	4/1.48	30	32.14	23.86
ParcPrinter	30	9	16.00	0.06	3/1.28	30	15.67	0.01
Parking	20	17	39.50	38.84	2/1.14	2	68.00	686.72
Pegsol	30	6	16.00	1.71	4/1.09	30	16.17	0.06
Pipes-NonTan	50	45	26.36	3.23	3/1.62	25	113.84	68.42
Rovers	40	27	38.47	108.59	2/1.39	20	67.63	148.34
Sokoban	30	3	80.67	7.83	3/2.58	23	166.67	14.30
Storage	30	25	12.62	0.06	2/1.48	16	29.56	8.52
Tidybot	20	7	42.00	532.27	3/1.81	16	70.29	184.77
Transport	30	21	54.53	94.61	2/2.00	17	70.82	70.05
Visitall	20	19	199.00	0.91	1/1.00	3	2485.00	174.87
Woodworking	30	30	21.50	6.26	2/1.07	12	42.50	81.02
...								
Summary	1150	819	44.4	55.01	2.5/1.6	789	137.0	91.05

Why IW does so well? A Width Notion

Consider a **chain** $t_0 \rightarrow t_1 \rightarrow \ldots \rightarrow t_n$ where each t_i is a **set of atoms** from P

- A chain is **valid** if t_0 is true in Init and **all optimal plans** for t_i can be **extended into optimal plans** for t_{i+1} by adding a **single** action
- The **size** of the chain is the **size of largest** t_i in the chain
- **Width** of P is **size of smallest** chain $t_0 \rightarrow t_1 \rightarrow \ldots \rightarrow t_n$ such that that the optimal plans for t_n are optimal plans for P.

Theorem 1: Domains like Blocks, Logistics, Gripper, . . . have all **bounded and small width**, independent of problem **size** provided that goals are **single atoms**

Theorem 2: **IW** runs in time exponential in width of P

IW is **blind search**. It doesn't use PDDL, can **plan effectively** with a **simulator**

IW on the Atari Video Games

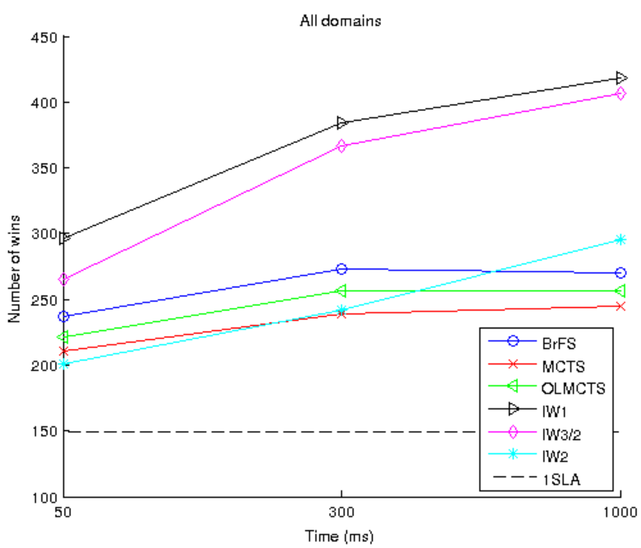
	IW(1)		2BFS		BrFS	UCT
Game	Score	Time	Score	Time	Score	Score
Alien	**25634**	81	12252	81	784	7785
Amidar	**1377**	28	1090	37	5	180
Assault	953	18	827	25	414	**1512**
Asterix	153400	24	77200	27	2136	**290700**
Asteroids	**51338**	66	22168	65	3127	4661
Atlantis	159420	13	154180	71	30460	**193858**
Bank Heist	**717**	39	362	64	22	498
Battle Zone	11600	86	**330800**	87	6313	70333
Beam Rider	9108	23	**9298**	29	694	6625
. . .	. . .	. . .	. . .	. . .	. . .	. . .
Robot Tank	**68**	34	52	34	2	50
Seaquest	**14272**	25	6138	33	288	5132
Space Invaders	2877	21	**3974**	34	112	2718
Star Gunner	1540	19	**4660**	18	1345	1207
Tennis	**24**	21	**24**	36	-24	3
Time Pilot	35000	9	36180	29	4064	**63855**
Tutankham	172	15	204	34	64	**226**
Up And Down	**110036**	12	54820	14	746	74474
Venture	**1200**	22	980	35	0	0
Video Pinball	**388712**	43	62075	43	55567	254748
Wizard Of Wor	**121060**	25	81500	27	3309	105500
Zaxxon	**29240**	34	15680	31	0	22610

# Times Best (54 games)	**26**	13	1	19

Avg Score collected by IW(1) vs. UCT and other when used in on-line mode (lookahead) in 54 Games. **Atoms** = values of each of the 128 **bytes** in 1024-bit state (Lipovetzky et. al. 2015)

IW on the General-Video Games (GVG-AI)

Time	50ms				300ms					
Game	**BrFS**	MC	OLMC	IW(1)	**BrFS**	MC	OLMC	IW(1)	1-Look	RND
Camel Race	**2**	1	1	0	1	3	0	**24**	0	1
Digdug	0	0	0	0	0	0	0	0	0	0
Firestorms	12	6	2	**13**	14	7	6	**25**	10	0
Infection	20	21	19	**22**	21	19	**22**	21	19	22
Firecaster	0	0	0	0	0	0	**1**	0	0	0
Overload	9	6	8	**20**	17	3	5	**23**	0	0
Pacman	1	0	0	**2**	1	1	4	**14**	0	0
Seaquest	13	13	**15**	9	11	17	**22**	9	12	0
Whackamole	20	18	**25**	23	22	23	**25**	21	21	5
Eggomania	0	0	1	**21**	0	0	2	**22**	0	0
Total	77	65	71	**110**	87	73	87	**159**	62	28

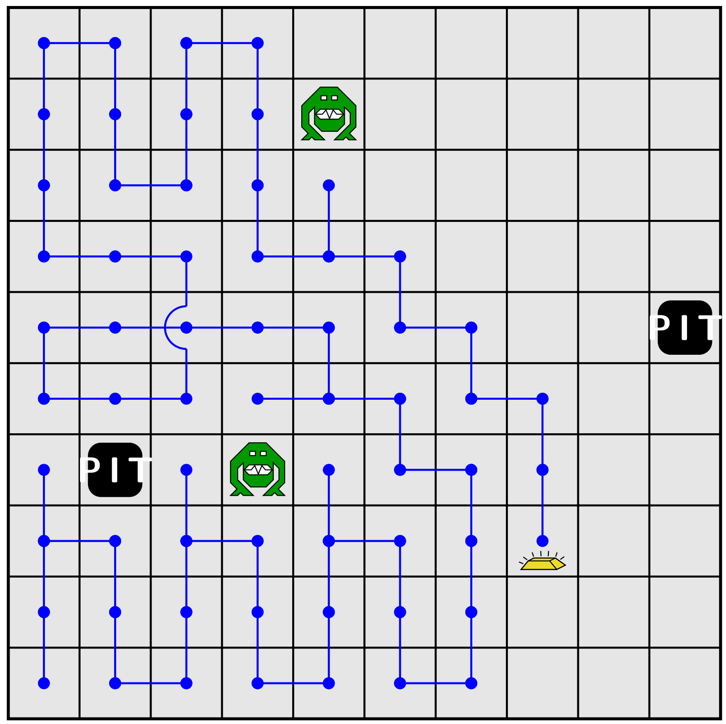


Top: # wins per game out of 25

Left: # wins as function of time for diff algorithms (T. Geffner and G. 2015)

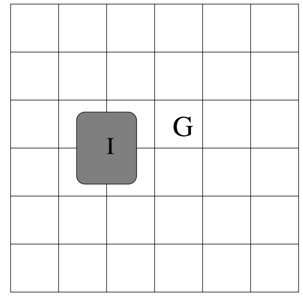
Something Different: Planning with Partial Feedback

How to **act** and **scale up** in the wumpus world?



Number of states $\approx 100^2 \times 3^{100}$. Number of belief states exponential in that number

Start Simple: Conformant Planning



- call a **set** of possible states, a **belief state**
- actions then map a belief state b into a bel state $b_a = \{s' \,|s' \in F(a,s) \;\&\; s \in b\}$
- **conformant problem** becomes a path-finding problem in **belief space**

Problem: number of belief state is **doubly exponential** in number of variables.

– **effective representation** of belief states b

– **effective heuristic** $h(b)$ for estimating cost in belief space

Alternative: translate into classical planning (Palacios & G, JAIR-2009)

Basic Translation K_0: Conformant into Classical

Given **conformant problem** $P = \langle F, O, I, G \rangle$

- F stands for the fluents in P
- O for the operators with effects $C \rightarrow L$
- I for the initial situation (**clauses** over F-literals)
- G for the goal situation (set of F-literals)

Define **classical problem** $K_0(P) = \langle F', O', I', G' \rangle$ as

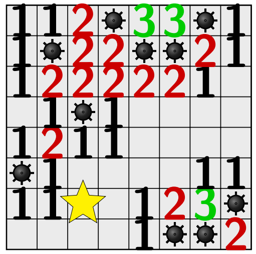
- $F' = \{KL, K\neg L \mid L \in F\}$
- $I' = \{KL \mid$ clause $L \in I\}$
- $G' = \{KL \mid L \in G\}$
- $O' = O$ but preconds L replaced by KL, and effects $C \rightarrow L$ replaced by $KC \rightarrow KL$ (**supports**) and $\neg K \neg C \rightarrow \neg K \neg L$ (**cancellation**)

$K_0(P)$ is **sound** but **incomplete**: classical plans that solve $K_0(P)$ solve P but not vice versa. **Complete** translations K_i **exponential** in **width** parameter, yet . . .

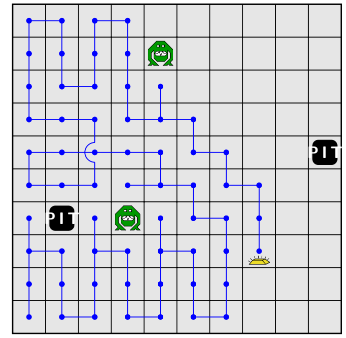
Using Classical Planners for Planning with Sensing

- A **partially observable** problem $P = \langle F, O, I, G, M \rangle$ is a **conformant** problem $P' = \langle F, O, I, G \rangle$ extended with a **sensor model** M:
  - ▷ M = set of **sensors** (C, L): if C true, value of L **observable**
- Define **optimistic relaxation** $K(P)$ as $K_0(P) = \langle F', O', I', G' \rangle$ extended with **extra actions** for **invariants** and **sensors**:
  - ▷ $O_{inv} = \{KC \rightarrow K\neg L$ for **invariant clauses** $C \rightarrow L$ in $I\}$
  - ▷ $O_{sen} = \{KC \wedge \neg KL \wedge \neg K\neg L \rightarrow KL\ ,\ KC \wedge \neg KL \wedge \neg KL \rightarrow K\neg L$ for **sensors** (C, L) in $M\}$
- Use $K(P)$ for **on-line partially observable planner** (Bonet and G., 2011, 2014)
  - ▷ **Action Selection: Classical plan** from $K(P)$ **executed** until **actual** observations refute assumptions; then **replan. Beliefs** tracked in KL literals
- **Exploitation or exploration** principle ensures that for **width-1** problems with **no dead-ends**, process **always reaches goal**

Empirical Results



Minesweeper



Wumpus Problem

domain	problem	#sim	solved	average calls	average length	avg. time in seconds total	avg. time in seconds prep	avg. time in seconds exec
mines	4x4	100	35	5.1	18.0	11.3	10.7	0.6
mines	6x6	100	37	9.6	38.0	522.4	506.6	15.8
mines	8x8	100	43	13.1	66.0	3488.2	3365.4	122.7
wumpus	5x5	100	100	12.2	15.2	1.4	0.9	0.4
wumpus	10x10	100	100	54.1	60.5	182.5	173.2	9.2
wumpus	15x15	100	100	109.7	121.0	3210.3	3140.3	70.0

E.g., in 15x15 Wumpus: 100% instances solved; 0.57 secs per action in execution

Last Theme: Planning with Nested Beliefs

- Belief tracking in **partially observable** planning is simple (semantically)

  ▷ **Beliefs** are sets of states (or probability distributions)
  ▷ If b is belief before action a, belief b_a **after action** is:

$$b_a = \{s' \mid s' \in F(a, s) \;\&\; s \in b\}$$

  ▷ If then observation o is obtained, belief b_a^o **after observation** is:

$$b_a^o = \{s' \mid s' \in b_a \text{ and } o \in O(s, a)\}$$

- Agent **knows** p if p is true in all states s in current belief b

- Belief tracking in **presence of other agents** more complicated but required for **communication**

Example: Communication as Planning

1	**2**	3	4

- **Initially** agents A and B at 2, and some blocks b_i not at 2
- **Goal:** A **knows** where b_1 is and B **knows** where b_2 is
- **Actions:** agents can **move**, **communicate**, and **sense** blocks in room
- Key questions: **what to sense** and **what to communicate**; shortest plan is:
  - ▷ A **moves** left to 1
  - ▷ B **moves** right to 3
  - ▷ A **senses** which blocks are in 1
  - ▷ B **senses** which blocks are in 3
  - ▷ A **tells** B whether b_2 in 1
  - ▷ B **tells** A whether b_1 in 3
- Knowing **what to communicate** and **when**, requires modeling **nested beliefs**; e.g., B knows that A knows where b_1 is after plan, else it'd go and sense 4

Beliefs in Multiagent Agent Settings

- Beliefs not only about the **world** but **about beliefs of other agents**
- E.g., $K_1K_2p \wedge K_1\neg K_3p$ says that 1 knows that 2 knows p and that 3 doesn't
- Such formulas cannot be evaluated in beliefs represented by **sets of states** (truth valuations)
- Futher **structure** required:
  - ▷ **Kripke structure** $\mathcal{K} = \langle W, R, V \rangle$ where W is set of **worlds** w, R is a set of **accessibility relations** R^i on worlds, one for each agent i, and $V(w)$ is **truth valuation** for world w
  - ▷ For **objective** formula A, $\mathcal{K}, w \models K_iA$ **iff** A is true in $V(w)$
  - ▷ For **epistemic** formula K_iA, $\mathcal{K}, w \models K_iA$ **iff** $\mathcal{K}, w' \models A$ for all w' s.t. $R^i(w, w')$
- **Questions:**
  - ▷ How to **specify** Kripke structures encoding **initial beliefs**?
  - ▷ How to **update** them as **actions** are applied and **observations** gathered?

A Basic "STRIPS" Solution to Multiagent Beliefs

- Agents assumed to start with **common initial belief** about the world given by **set of states** S_0
- Agents act on the world, sense environment, and **sense beliefs of other agents**
- Such events are assumed to be **public**
- This results in **unique** Kripke structure $\mathcal{K}_t = \langle W_t, R_t, V_t \rangle$ for each time step t:
  - ▷ $W_t = S_0$; i.e., worlds associated with the possible initial states in S_0,
  - ▷ $V_t(s_0)$ is the state that results from s_0 after the actions done up to time t,
  - ▷ $R^i_t(s_0, s'_0)$ true unless agent i **sensed** A at $t' < t$ and $\mathcal{K}_{t'}, s_0 \models A$ and $\mathcal{K}_{t'}, s'_0 \models \neg A$
- The problem P of finding a sequence of **actions**, **sensing**, and **communication acts** for achieving a goal G, can be **translated** into a **classical planning** problem $K(P)$, **solved** by **off-the-shelf** planners
- **Size** of the translation is **quadratic** in $|S_0|$ (Kominis and G. 2015)

Challenges and Opportunities in Planning

- **Technical Challenges**
  - ▷ **Scaling up** in **probabilistic** partially obs problems (POMDPs)
  - ▷ **Learning models**: how to act when action and sensor not fully known
  - ▷ **Learning states:** learning models from streams of actions and observations
  - ▷ **Hierarchies:** what to abstract away and when, **scaffolding**
  - ▷ **Multiagent**: generation and recognition of intentional behaviour
  - ▷ **Constraints:** like **geometrical constraints** in **motion planning**

- **Applications**
  - ▷ robotics, video-games, dialogue, interaction, . . .

- **Cognitive Science**
  - ▷ derivation of **heuristics** provides model for quick global **appraisals**
  - ▷ scalability and computation as **sources of insight**

Summary

- **Planning** is model-based approach to **autonomous behavior**
- **Planning models** come in many forms: uncertainty, feedback, costs, . . .
- Key technique in **classical planning** is automatic derivation and use of **heuristics**
- Yet simple **blind search** algorithms like **IW** can perform well too and wider scope (Atari Games)
- Power of classical planners used for other tasks via **transformations:**
  - ▷ **on-line planning with partial observability**
  - ▷ **planning with nested beliefs when other agents present**
  - ▷ . . .
- **Structure: width-notions** for classical planning, belief tracking, reductions, . . .