Reduccion de la Planificacion Conformante a SAT mediante Compilacion a d–DNNF

Héctor Palacios
UPF

Héctor Geffner
ICREA/UPF

Planning

- Agent performs **actions** to achieve a **goal**
- Many flavors: uncertainty, time, resources, etc
- Last decade: shift from **theoretical** to **empirical** based. significant improvement

Planning

- Agent performs **actions** to achieve a **goal**
- Many flavors: uncertainty, time, resources, etc
- Last decade: shift from **theoretical** to **empirical** based. significant improvement
- *Classical Planning*: simplest flavor

  From **a** initial state, reach a goal by doing a plan (**sequence** of actions)

  Example: Robot navigation: starts from a position, has a map

Planning

- Agent performs **actions** to achieve a **goal**
- Many flavors: uncertainty, time, resources, etc
- Last decade: shift from **theoretical** to **empirical** based. significant improvement
- *Classical Planning*: simplest flavor

  From **a** initial state, reach a goal by doing a plan (**sequence** of actions)

  Example: Robot navigation: starts from a position, has a map
- *Conformant Planning*: slight uncertainty

  Many possible initial states: **one** plan working for **every** initial state

  Example: a blind Robot has a map, but doesn't know its initial position

Motivation

- Classical Planning as SAT
  - Obtain a formula from a problem, call a **solver**
  - Very successful!

Motivation

- Classical Planning as SAT
  - Obtain a formula from a problem, call a **solver**
  - Very successful!
- Conformant Planning is NP-hard: can't be mapped to **one** SAT
  - We want a **formula** to feed a SAT solver
  - **Obtaining** can be expensive

Motivation

- Classical Planning as SAT
  - Obtain a formula from a problem, call a **solver**
  - Very successful!
- Conformant Planning is NP-hard: can't be mapped to **one** SAT
  - We want a **formula** to feed a SAT solver
  - **Obtaining** can be expensive
- We present a **optimal conformant planner**: obtain a formula, SAT
- The planner just need *two off-the-shelf components:*

  a knowledge compiler and a SAT solver

  No specific search algorithm!

Outline

- Classical Planning as SAT
- Conformant Planning as SAT
- A propositional formula for solving Conformant Planning as SAT
- Knowledge Compilation to generate the formula
- Algorithm
- Experiments
- Discussion
- Summary

Classical Planning

- States: set of **fluents variables** describing the situation
- Discrete time
- **One** initial state, goal states
- Apply action a
  - requires **precondition**(a) $\bigwedge$
  - guarantee **effect**(a) in the next time step

Classical Planning

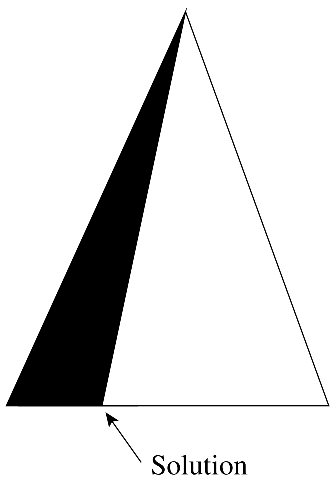
- States: set of **fluents variables** describing the situation
- Discrete time
- **One** initial state, goal states
- Apply action a
  - requires **precondition**(a) $\bigwedge$
  - guarantee **effect**(a) in the next time step

Example: Robot Navigation

- State consist of fluents: horizontal position, vertical position
- Actions: move-up, move-left

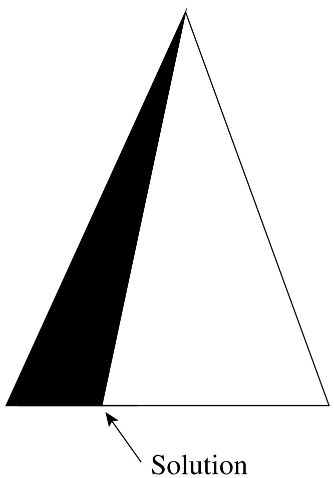
Classical Planning: Complexity and Solution

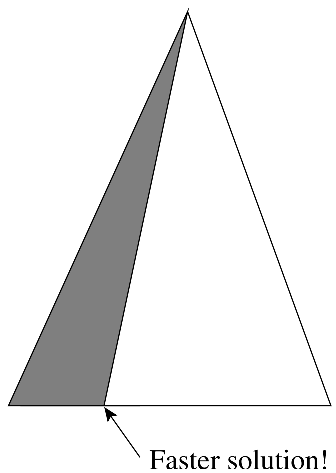
- NP-complete (as SAT, exponential) assuming fixed horizon



Classical Planning: Complexity and Solution

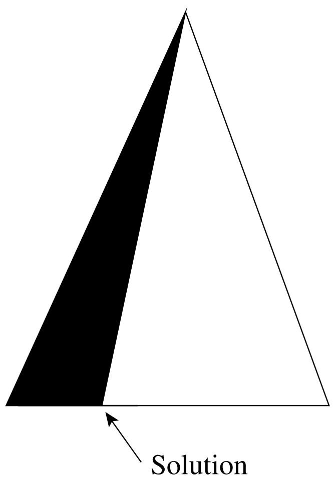
- NP-complete (as SAT, exponential) assuming fixed horizon
- SAT solvers do well in many cases.

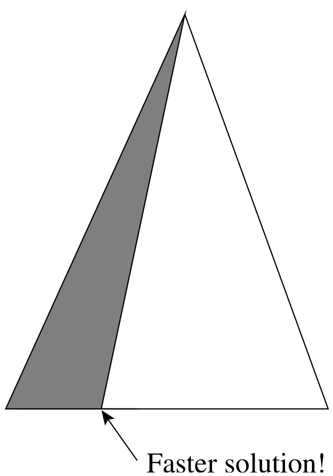




Classical Planning: Complexity and Solution

- NP-complete (as SAT, exponential) assuming fixed horizon
- SAT solvers do well in many cases.





- To map the *decision problem* of classical planning, horizon k to SAT
  - For k, **generate** a propositional theory Φ **encoding** the problem
  - If Φ is SAT, report a solution

Classical Planning as SAT

- A propositional theory Φ **encoding** the problem, for horizon k
  - A variable for **every** action and fluent at **every** time step: a_i, f_i
  - Describe **relation** between actions and fluents in time

    Example: $\textsf{MOVE-LEFT}_1 \wedge \textsf{POS-HORIZ}_1{=}3 \supset \textsf{POS-HORIZ}_2{=}2$
  - Ensure that **models** of Φ are *all* the *sound* **executions**
- Call a SAT solver over Φ

Classical Planning as SAT

- A propositional theory Φ **encoding** the problem, for horizon k
  - A variable for **every** action and fluent at **every** time step: a_i, f_i
  - Describe **relation** between actions and fluents in time

    Example: MOVE-LEFT$_1$ $\wedge$ POS-HORIZ$_1$=3 $\supset$ POS-HORIZ$_2$=2
  - Ensure that **models** of Φ are *all* the *sound* **executions**
- Call a SAT solver over Φ

Example:

- Problem with fluents $\{p, q\}$ and actions $\{a\}$
- Vars of Φ (k = 2): $\{p_0, q_0, \quad a_0, \quad p_1, q_1, \quad a_1, \quad p_2, q_2\}$

Conformant Planning SAT

- Classical planning + **many** possible initial states
- Logical theory Φ:

  same + logical description of initial states

Conformant Planning SAT

- Classical planning + **many** possible initial states
- Logical theory Φ:

  same + logical description of initial states

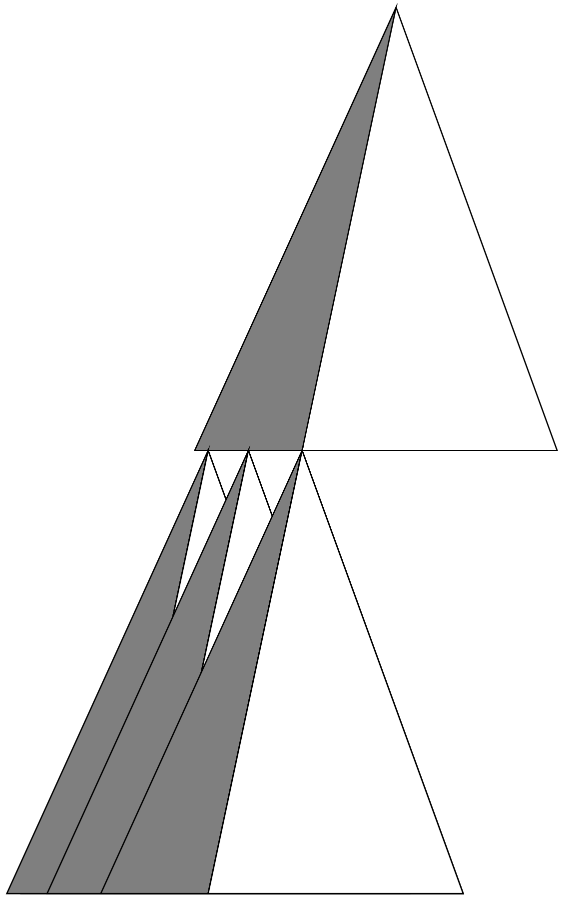
  - Models: plans for **one** initial state (optimistic)
  - We want **one** plan for **all** initial states (pessimistic)

Conformant Planning SAT

- Classical planning + **many** possible initial states
- Logical theory Φ:

  same + logical description of initial states

  - Models: plans for **one** initial state (optimistic)
  - We want **one** plan for **all** initial states (pessimistic)
- Naive solution
  - Start from horizon $k = 0$, until find a solution
    - For k, **generate** a propositional theory Φ **encoding** the problem
    - Generate **candidate** (SAT) and Test it (SAT)



A propositional formula for Conformant Planning

- For a **specific** s_0, the plans are the models of

$$T + s_0 \qquad\qquad \text{as in classical planning}$$

A propositional formula for Conformant Planning

- For a **specific** s_0, the plans are the models of

  $T + s_0$ as in classical planning

- Plans conformant for **all** s_0, are the models of?

$$\bigwedge_{s_0 \in \mathit{Init}} T + s_0$$

A propositional formula for Conformant Planning

- For a **specific** s_0, the plans are the models of

  $T + s_0$ as in classical planning

- Plans conformant for **all** s_0, are the models of?

  $$\bigwedge_{s_0 \in \mathit{Init}} T + s_0$$

  No: **same plan**, **different** executions

A propositional formula for Conformant Planning

- For a **specific** s_0, the plans are the models of

  $T + s_0$ as in classical planning

- Plans conformant for **all** s_0, are the models of?

  $$\bigwedge_{s_0 \in \mathit{Init}} T + s_0$$

  No: **same plan**, **different** executions

- **Project** over actions: models of T but **only** over actions

  project$(a \wedge b, \{a\}) = a$, project$((a \wedge b) \vee c, \{a, c\}) = a \vee c$

A propositional formula for Conformant Planning

- For a **specific** s_0, the plans are the models of

  $T + s_0$ as in classical planning

- Plans conformant for **all** s_0, are the models of?

  $$\bigwedge_{s_0 \in \mathit{Init}} T + s_0$$

  No: **same plan**, **different** executions

- **Project** over actions: models of T but **only** over actions

  $\text{project}(a \wedge b, \{a\}) = a, \quad \text{project}((a \wedge b) \vee c, \{a, c\}) = a \vee c$

- **Theorem**: *The conformant plans are the Models of*

  $$\bigwedge_{s_0 \in \mathit{Init}} \text{project}[\, T + s_0 \; ; \; \mathit{Actions} \,]$$

Conformant Planning(horizon k)

1. **Generate** theory T for horizon k
2. **Construct** the formula T_{cf} where

$$T_{\text{cf}} \quad = \quad \bigwedge_{s_0 \in \mathit{Init}} \text{project}[\, T + s_0 \,;\, \mathit{Actions} \,]$$

3. Obtain a **Plan** by calling *once* a **SAT** solver over T_{cf}

Conformant Planning(horizon k)

1. **Generate** theory T for horizon k

2. **Construct** the formula T_{cf} where

$$T_{\text{cf}} \quad = \quad \bigwedge_{s_0 \in \textit{Init}} \text{project}[\, T \,+\, s_0 \,;\, \textit{Actions} \,]$$

3. Obtain a **Plan** by calling *once* a **SAT** solver over T_{cf}

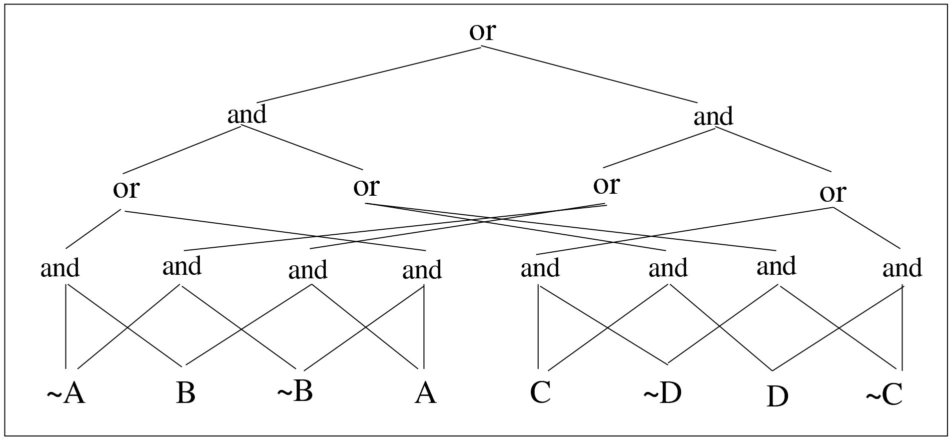
if we **can** do projection and conditioning $(T + s_0)$

Answer: Knowledge compilation

- **Transform** a theory to a target language, **expensive** (exponential), then make **cheap** operations

Answer: Knowledge compilation

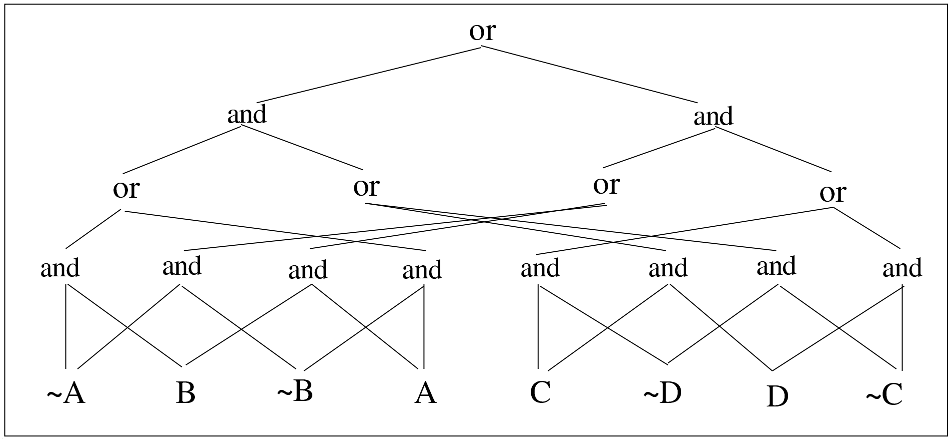
- **Transform** a theory to a target language, **expensive** (exponential), then make **cheap** operations
- We use *deterministic - Decomposable Negation Normal Form*, *d*–DNNF, a form akin to OBDDs



- Supports **poly-time conditioning and projection**

Answer: Knowledge compilation

- **Transform** a theory to a target language, **expensive** (exponential), then make **cheap** operations
- We use *deterministic - Decomposable Negation Normal Form*, *d*–DNNF, a form akin to OBDDs
- Supports **poly-time conditioning and projection**



- Some OBDDs are **exponentially larger** than their equivalent d–DNNFs
- **Public libraries** for compilation from CNF to OBDDs or d–DNNFs

Conformant Planning as SAT

Start from horizon $k = 0$ increasing until find a solution

1. **Generate** theory T for horizon k
2. T is **compiled** *(*once) into a d–DNNF theory T_{c}
3. From T_{c}, the transformed theory

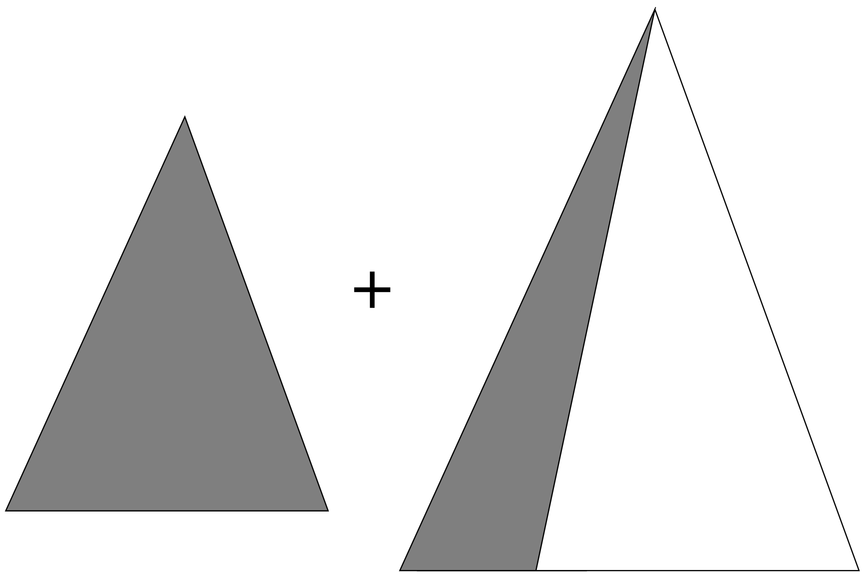
$$T_{\text{cf}} \quad = \quad \bigwedge_{s_0 \in \mathit{Init}} \text{project}[\, T_{\text{c}} + s_0 \,;\, \mathit{Actions} \,]$$

   is obtained by linear operations in T_{c}
4. A **SAT solver** is called *(*once) over T_{cf}

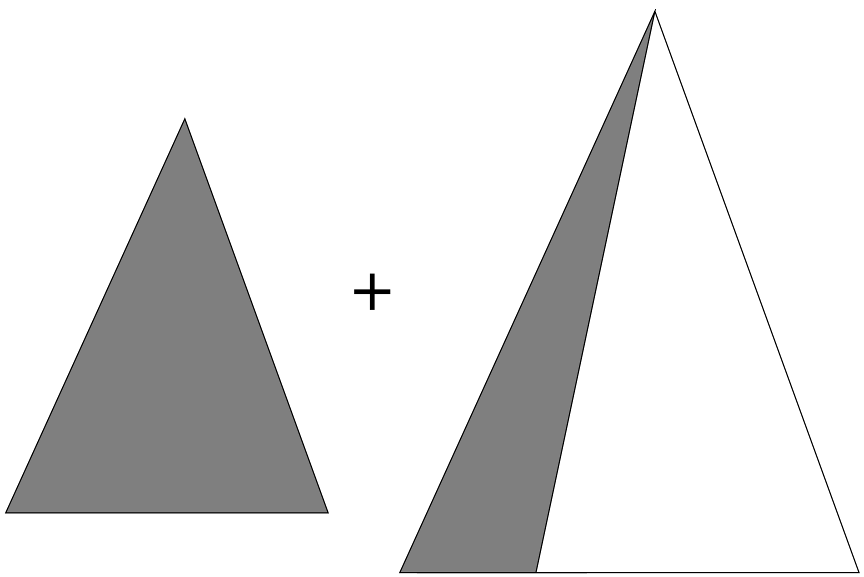
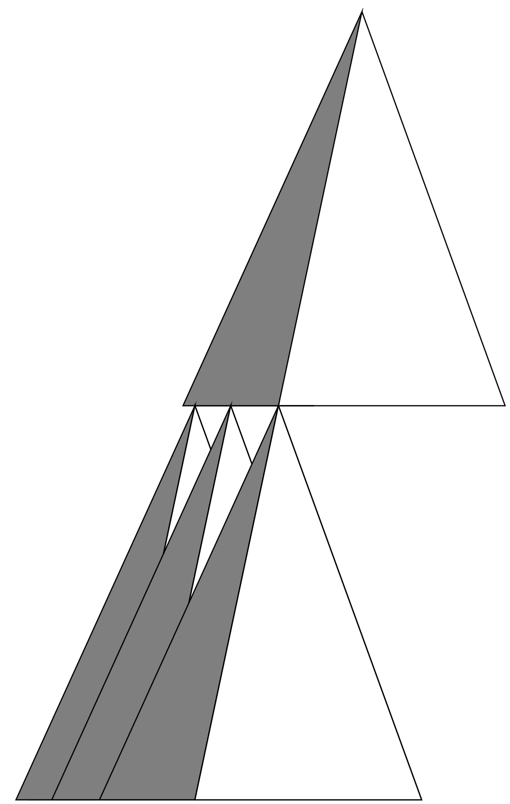
Require: a *compiler* and a *sat solver*: no specific search algorithm

For each horizon k

Compile & SAT approach



For each horizon k

Compile & SAT approach	Naive approach
	

Problems

Ring n rooms arranged in a circle. A robot can move one step a time. The room features **windows** that can be **closed** and **locked**. Initially, the position of the robot and the status of the windows is not known

Square Center A robot without sensors can move in a **grid** north, south, east, and west, and its goal is to **get to the middle** of the room. The size of the grid is $2^n \times 2^n$

Sorting networks Build a circuit made of **compare-and-swap** gates that **maps** an input vector of n boolean variables **into** the corresponding **sorted vector**

Compile time

		CNF(T)		d–DNNF T_c			CNF(T_{cf})	
problem	N^*	vars	clauses	nodes	edges	time	vars	clauses
ring-r7	20	1081	3683	1008806	2179064	192.2	976203	3105362
ring-r8	23	1404	4814	3887058	8340295	1177.1	3779477	11957085
sq-center-e3	20	976	3642	11566	22081	1.1	9664	27956
sq-center-e4	44	4256	16586	90042	174781	47.1	81404	238940
sort-s7	16	1484	6679	115258	283278	12.4	112756	390997
sort-s8	19	2316	12364	363080	895247	77.2	359065	1246236

- **Exponential** increasing because compilation
- **Linear** translation from d–DNNF to CNF
- Big theories do not imply **hard** problems
- Compilation is **not** the bottleneck

d–DNNF compiler by Adnan Darwiche

Search time

	problem	N^*	$\#S_0$	sat call with horiz N^* time	decisions	#act	sc with horizon $N^* - 1$ time	decisions
Serial	ring-r7	20	15309	$^\circ$ 2.1	2	20	$^\circ$ 0.8	0
	ring-r8	23	52488	$>$ 1.8Gb			$^\circ$ 2.4	0
	sq-center-e3	20	64	18.8	52037	20	207.4	207497
	sq-center-e4	44	256	5184.4	1096858	44	$> 2h$	
	sort-s6	12	64	40.0	34451	12	$> 2h$	
	sort-s7	16	128	3035.6	525256	16	$> 2h$	
	sort-s8	19	256	$> 2h$			$> 2h$	
Parallel	sq-center-e4	22	256	423.1	244085	44	1181.5	439532
	sort-s7	6	128	46.1	18932	18	355.4	48264
	sort-s8	6	256	$^\circ$ 4256.6	533822	23	$> 2h$	

SAT solver: (SIEGE_V4 or *zChaff*). Time in seconds.

Blue: our model-counting based planner couldn't solve it (ICAPS'05)

Comparison with other works

- No many optimal conformant planners, but many suboptimal
- In general, **better** on very difficult problems: sort, cube
- **Worst** in problems close to classical planning (less uncertainty) or many objects. Ex: bomb in the toilet with 100 bombs

Discussion

- Our theories are easy to compile following their **stratified structure**: fluents f_i are related with other fluents f_i and actions a_i and a_{i-1}
- Without this, compiling using the **stratification** vs. an **automatic strategy** of the compiler.
  - sort-7-ser: 12s vs 40s. Automatic: double size of the graph
  - sq-center-4: 43.9s vs $>$ 2 hours

Discussion

- Our theories are easy to compile following their **stratified structure**: fluents f_i are related with other fluents f_i and actions a_i and a_{i-1}
- Without this, compiling using the **stratification** vs. an **automatic strategy** of the compiler.
  - sort-7-ser: 12s vs 40s. Automatic: double size of the graph
  - sq-center-4: 43.9s vs $>$ 2 hours
- Compilation **too** expensive for problems with **many** objects, but they are solved easily by others
- Other ways to project? renaming

Summary

- **Conformant Planning**: *slight* variation of classical planning, relevant for insight in other flavors with **uncertainty**
- **Main contribution**: propositional formula for conforman planning
- To solve a problem, **one** compiler call and **one** SAT call until k optimal
  - **Simple** and powerful scheme
- **Encouraging** results
- Compilation is **not the bottleneck**
- Some instance **haven't been** solved before (sort, cube...)
- Lot of improvement on problems close to **classical planning**

Acknowledgement

- Blai Bonet: code for parsing the PDDL problem specification and generation of CNF and previous join work
- Adnan Darwiche: compiler from CNF to d–DNNF and previous joint work
- Reviewers

thank you!

Conformant Planning Theory

Slight variation of encoding in SATPLAN

1. **Init:** a clause C_0 for each init clause $C \in I$.
2. **Goal:** a clause C_N for each goal clause $C \in G$.
3. **Actions:** For $i = 0, 1, \ldots, N-1$ and $a \in O$:

$$a_i \supset \text{pre}(a)_i \quad \text{(precondition}$$

$$\text{cond}^k(a)_i \wedge a_i \supset \text{effect}^k(a)_{i+1}, \quad k = 1, \ldots, k_a \quad \text{(effects)}$$

4. **Frame:** for $i = 0, 1, \ldots, N-1$, each fluent literal

$$l_i \wedge \bigwedge_{\text{cond}^k(a)} \neg[\text{cond}^k(a)_i \wedge a_i] \supset l_{i+1}$$

   where the conjunction ranges over the conditions $\text{cond}^k(a)$ associated with effects $\text{effect}^k(a)$ that support the complement of l.

5. **Exclusion:** $\neg a_i \vee \neg a'_i$ for $i = 0, \ldots, N-1$

Conformant Planning Theory: Example

Problem:

- Fluents: p, q, r
- Init: $p \vee q, \neg r$. Goal: r
- Actions
  - a_q: if p effect is q
  - a_r: if q effect is r

Theory Φ for horizon $k = 2$

- Init: $p_0 \vee q_0, \neg r_0$
- Goal: r_2
- exclusion: $a_q0 \otimes a_r0$

Conformant Planning Theory: Example

Problem:

- Fluents: p, q, r
- Init: $p \vee q, \neg r$. Goal: r
- Actions
  - a_q: if p effect is q
  - a_r: if q effect is r

Theory Φ for horizon $k = 2$

- Init: $p_0 \vee q_0, \neg r_0$
- Goal: r_2
- exclusion: $a_q 0 \otimes a_r 0$
- effects:

$$a_q 0 \wedge p_0 \supset q_1$$
$$a_r 0 \wedge q_0 \supset r_1$$

- frame, for each literal

p	$p_0 \supset p_1$
$\neg p$	$\neg p_0 \supset \neg p_1$
q	$\neg q_0 \supset \neg q_1$
$\neg q$	$\neg(a_q 0 \wedge p_0) \wedge \neg q_0 \supset \neg q_1$
r	$\neg r_0 \supset \neg r_1$
$\neg r$	$\neg(a_r 0 \wedge r_0) \wedge \neg r_0 \supset \neg r_1$

etc.

deterministic - Decomposable Negation Normal Form (d–DNNF)

- Normal form: NNF satisfying determinism and decomposability (see paper for details)
  - **Deterministic**: for each AND node, no variable appears in more than one conjunct
  - **Decomposable**: for each OR node, disjuncts are pairwise logically inconsistent
- Compiling to d–DNNF: a naive algorithm proceed doing **exhaustive** DPLL (all SAT)
- d-DNNF compilations are, typically, **exponentially** bigger
- Projection and conditioning are **lineal** in the size of the d-DNNF

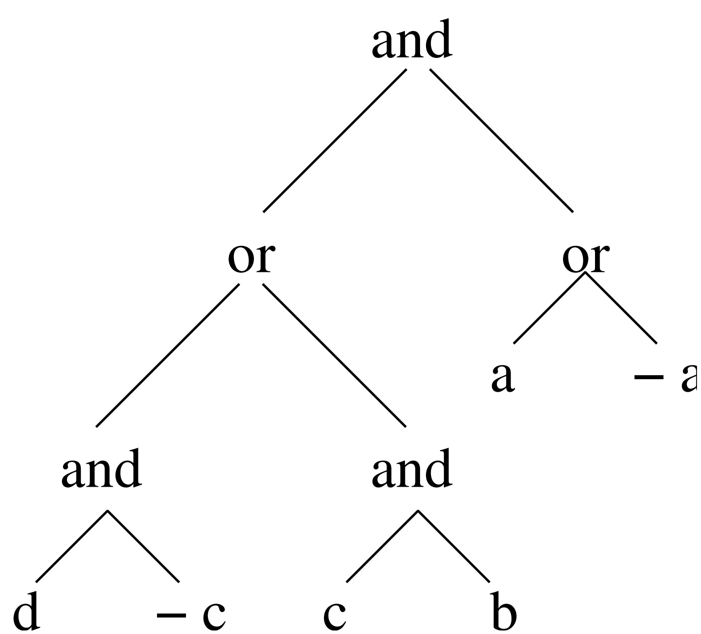
d-DNNF: Example

Theory

$a \vee \neg a$

$c \vee d$

$\neg c \vee b$



- **Decomposable?**

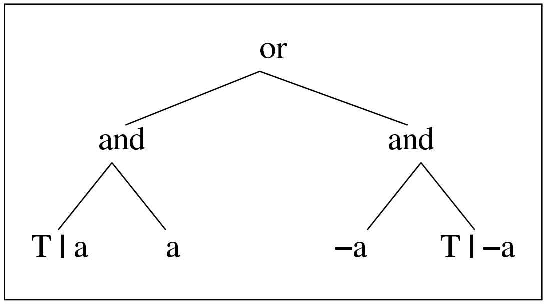
  For each OR node, disjuncts are pairwise logically inconsistent

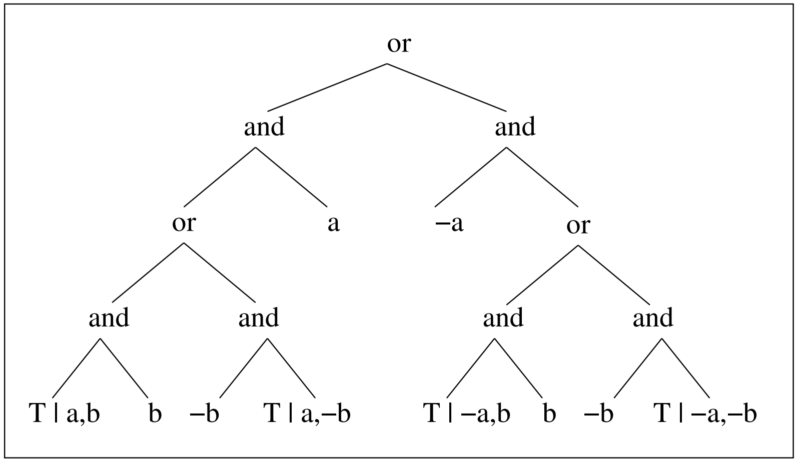
- **Deterministic?**

  For each AND node, no variable appears in more than one conjunct

Calculating the CNF efficiently

- We can ask the compiler to give the d–DNNF
  - Projected over actions and vars(s_0) (no fluents $i > 0$)
  - Make cases analysis **first** over vars(s_0)
- Then project$[\,T + s_0\,;\,$*Actions*$\,]$ can be **extracted as a subgraph**

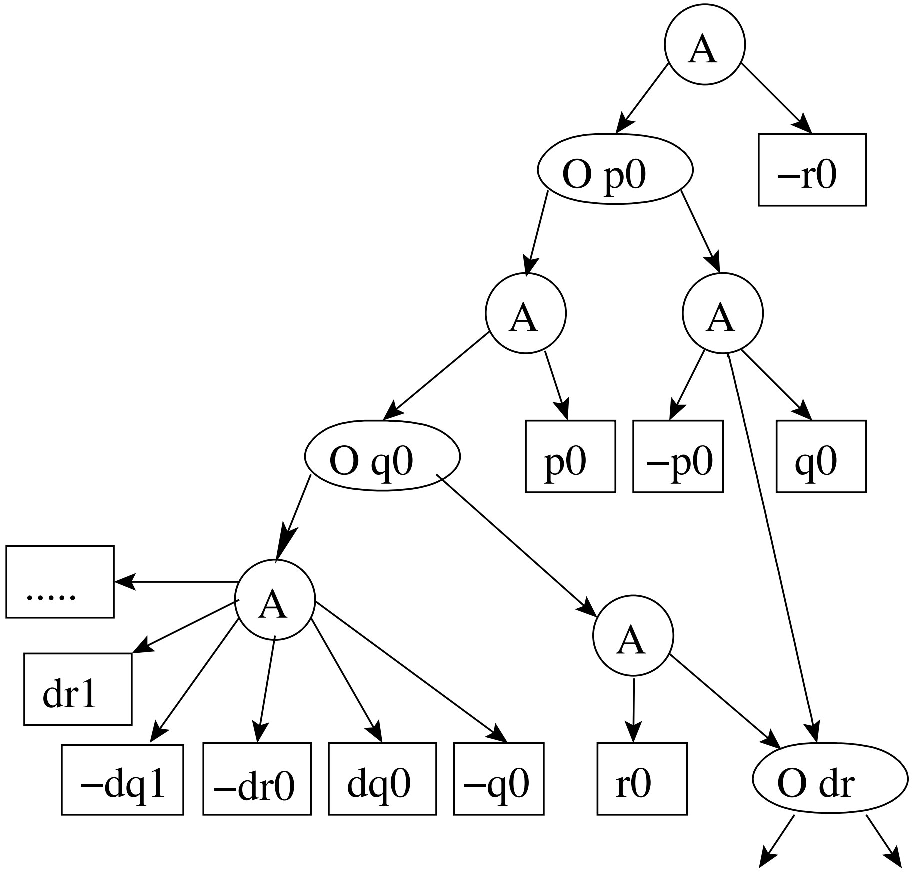




Then, we can construct $\bigwedge_{s_0 \in Init}$ project$[\,T + s_0\,;\,$*Actions*$\,]$
by making a **new graph** with the extracted subgraphs. Easy to CNF!

- Fluents: p, q, r
- Init: $p \vee q, \neg r$. Goal: r
- Actions:
  - a_q: if p effect is q
  - a_r: if q effect is r
- Solution: a_q, a_r

Compiling for $k = 2$ ...

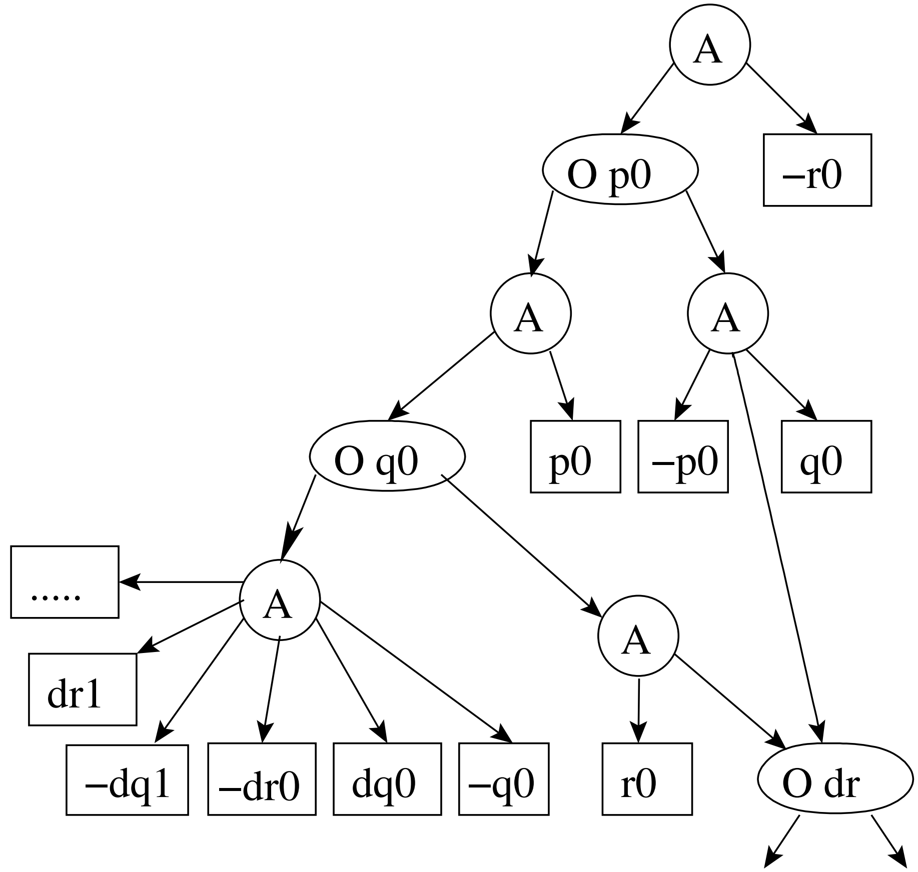


- Fluents: p, q, r
- Init: $p \vee q, \neg r$. Goal: r
- Actions:
  - a_q: if p effect is q
  - a_r: if q effect is r
- Solution: a_q, a_r

Compiling for $k = 2$ ...

Asking the compiler to:

- Make cases analysis **first** over init vars: p_0, q_0, r_0
- **Project while compiling** over init + action vars

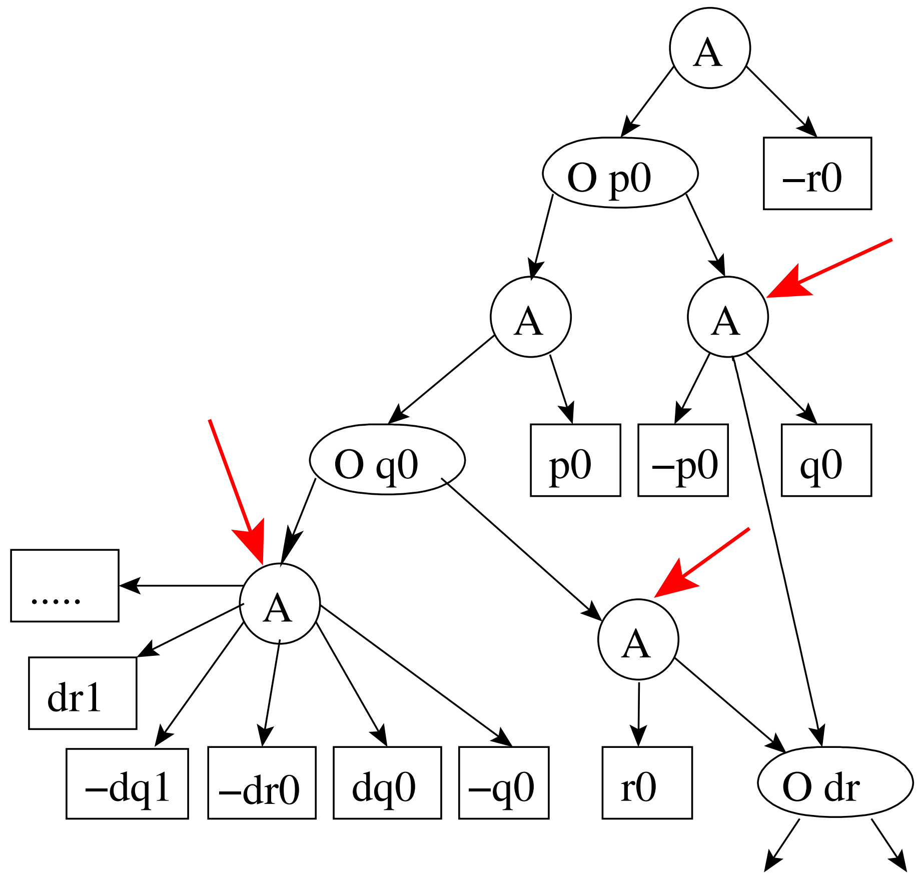


- Fluents: p, q, r
- Init: $p \vee q, \neg r$. Goal: r
- Actions:
  - a_q: if p effect is q
  - a_r: if q effect is r
- Solution: a_q, a_r

Compiling for $k = 2$ ...

Asking the compiler to:

- Make cases analysis **first** over init vars: p_0, q_0, r_0
- **Project while compiling** over init + action vars



Projection, a logical operation

- Don’t want to care about some variables

Projection, a logical operation

- Don't want to care about some variables
- Example: want to *forget* f_1 from $\phi = (a_1 \wedge f_1) \vee a_2$

Projection, a logical operation

- Don't want to care about some variables
- Example: want to *forget* f_1 from $\phi = (a_1 \wedge f_1) \vee a_2$

$$
\begin{aligned}
\text{project}[\,\phi;\ \{a_1, a_2\}\,] \quad &= \quad \exists f_1\, \phi \\
&= \quad (\phi\,|\,f_1 = \text{true}) \ \vee \ (\phi\,|\,f_1 = \text{false}) \\
&= \quad ((a_1 \wedge \text{true}) \vee a_2) \ \vee \\
&\qquad ((a_1 \wedge \text{false}) \vee a_2) \\
&= \quad (a_1 \vee a_2)
\end{aligned}
$$

Models of $\phi = (a_1 \wedge f_1) \vee a_2$, if we **don't care** about f_1, are the models of $a_1 \vee a_2$

Projection, a logical operation

- Don't want to care about some variables
- Example: want to *forget* f_1 from $\phi = (a_1 \wedge f_1) \vee a_2$

$$
\begin{aligned}
\text{project}[\,\phi;\ \{a_1, a_2\}\,] \quad &= \quad \exists f_1\, \phi \\
&= \quad (\phi \,|\, f_1 = \text{true}) \ \vee \ (\phi \,|\, f_1 = \text{false}) \\
&= \quad ((a_1 \wedge \text{true}) \vee a_2) \ \vee \\
& \qquad ((a_1 \wedge \text{false}) \vee a_2) \\
&= \quad (a_1 \vee a_2)
\end{aligned}
$$

  Models of $\phi = (a_1 \wedge f_1) \vee a_2$, if we **don't care** about f_1, are the models of $a_1 \vee a_2$

- The **projection** of a formula over a subset of its variables is the **strongest** formula over those variables

Discussion (2)

- Conformant Planning can be solved as a QBF of the form solve

  $$\exists \textit{Plan}\, \forall s_0\, \exists \text{execution} \quad T$$

  Our method is **simple and generic**. Can be used to solve QBFs?

- Our CNFs theories are probably the biggest compiled to d-DNNF. Can we detect **stratified** structure in other CNFs?

- Relation with other problems that can't be map to SAT: all solutions to CNFs, unsat of CNFs, weighted CNF, maxSAT, MPE (Bay Nets).

- Further work: new theoretical notions for understanding the gap between theory and practice in SAT and CSP and beyond them: hypertree decomposition (chen & dalmau), semantic width (dechter), strong backdoors (gomes, selman).