Algorithms for Classical Planning

Jussi Rintanen

Beijing, IJCAI 2013
Planning
What to do to achieve your objectives?

- **Which actions** to take to achieve your objectives?
- **Number of agents**
  - single agent, perfect information: s-t-reachability in succinct graphs
  - + nondeterminism/adversary: and-or tree search
  - + partial observability: and-or search in the space of beliefs

**Time**
- asynchronous or instantaneous actions (integer time, unit duration)
- rational/real time, concurrency

**Objective**
- Reach a goal state.
- Maximize probability of reaching a goal state.
- Maximize (expected) rewards.
- temporal goals (e.g. LTL)
Planning
What to do to achieve your objectives?

- Which **actions** to take to achieve your objectives?
- Number of agents
  - single agent, perfect information: s-t-reachability in succinct graphs
  - + nondeterminism/adversary: and-or tree search
  - + partial observability: and-or search in the space of beliefs

**Time**
- asynchronous or instantaneous actions (integer time, unit duration)
- rational/real time, concurrency

**Objective**
- Reach a goal state.
- Maximize probability of reaching a goal state.
- Maximize (expected) rewards.
- temporal goals (e.g. LTL)
Planning
What to do to achieve your objectives?

- Which **actions** to take to achieve your objectives?
- Number of agents
  - single agent, perfect information: s-t-reachability in succinct graphs
  - + nondeterminism/adversary: and-or tree search
  - + partial observability: and-or search in the space of beliefs

**Time**
- asynchronous or instantaneous actions (integer time, unit duration)
- rational/real time, concurrency

**Objective**
- Reach a goal state.
- Maximize probability of reaching a goal state.
- Maximize (expected) rewards.
- temporal goals (e.g. LTL)
Hierarchy of Planning Problems

- POMDP (undecidable [MHC03])
- partially observable (2-EXPTIME [Rin04a])
- conditional/MDP (EXP [Lit97])
- temporal (EXPSPACE [Rin07])
- classical (PSPACE [GW83, Loz88, LB90, Byl94])
Classical (Deterministic, Sequential) Planning

- states and actions expressed in terms of state variables
- single initial state, that is known
- all actions deterministic
- actions taken sequentially, one at a time
- a goal state (expressed as a formula) reached in the end

Deciding whether a plan exists is PSPACE-complete.
With a polynomial bound on plan length, NP-complete.
Domain-Independent Planning

What is domain-independent?

- **general language** for representing problems (e.g. PDDL)
- **general algorithms** to solve problems expressed in it

Advantages and disadvantages:

+ Representation of problems at a high level
+ Fast prototyping
+ Often easy to modify and extend
  - Potentially high performance penalty w.r.t. specialized algorithms
  - Trade-off between generality and efficiency
Domain-Specific Planning

What is domain-specific?

- application-specific representation
- application-specific constraints/propagators
- application-specific heuristics

There are some planning systems that have aspects of these, but mostly this means: implement everything from scratch.
Domain-Dependent vs. -Independent Planning

Procedure

1. Formalize in PDDL
2. Try off-the-shelf planners
3. Problem solved?
   - Go domain-specific
   - Done
planning, diagnosis [SSL$^+$95], model-checking (verification)
How to Represent Planning Problems?

transition-based

constraint-based

planning

SMV
Petri Nets
Answer-Set Programs
PDDL
CSP
SAT

Introduction
State-Space Search
SAT
Symbolic search
Planners
Evaluation
References
PDDL - Planning Domain Description Language

- Defined in 1998 [McD98], with several extensions later.
- Lisp-style syntax
- Widely used in the planning community.
- Most basic version with Boolean state variables only.
- Action sets expressed as schemata instantiated with objects.

(:action analyze-2
  :parameters (?s1 ?s2 - segment ?c1 ?c2 - car)
  :precondition (and (CYCLE-2-WITH-ANALYSIS ?s1 ?s2)
                      (on ?c1 ?s1))
  :effect (and (not (on ?c1 ?s1))
              (on ?c2 ?s1)
              (analyzed ?c1)
              (increase (total-cost) 3)))
States are **valuations of state variables.**

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>State variables are</strong></td>
</tr>
<tr>
<td>LOCATION: {0, \ldots, 1000}</td>
</tr>
<tr>
<td>GEAR: {R, 1, 2, 3, 4, 5}</td>
</tr>
<tr>
<td>FUEL: {0, \ldots, 60}</td>
</tr>
<tr>
<td>SPEED: {-20, \ldots, 200}</td>
</tr>
<tr>
<td>DIRECTION: {0, \ldots, 359}</td>
</tr>
</tbody>
</table>
State-space transition graphs
Blocks world with three blocks
Actions
How values of state variables change

General form
precondition: A=1 ∧ C=1
effect: A := 0; B := 1; C := 0;

STRIPS representation
PRE: A, C
ADD: B
DEL: A, C

Petri net
Weaknesses in Existing Languages

- **High-level concepts** not easily/efficiently expressible. Examples: graph connectivity, transitive closure.
- Limited or no facilities to express **domain-specific** information (control, pruning, heuristics).
- The notion of classical planning is limited:
  - Real world rarely a single run of the sense-plan-act cycle.
  - Main issue often **uncertainty**, **costs**, or both.
  - Often **rational time** and concurrency are critical.
Formalization of Planning in This Tutorial

A problem instance in (classical) planning consists of the following.

- set $X$ of state variables
- set $A$ of actions $\langle p, e \rangle$ where
  - $p$ is the precondition (a set of literals over $X$)
  - $e$ is the effects (a set of literals over $X$)
- initial state $I : X \rightarrow \{0, 1\}$ (a valuation of $X$)
- goals $G$ (a set of literals over $X$)
The planning problem

An action $a = \langle p, e \rangle$ is applicable in state $s$ iff $s \models p$. The successor state $s' = \text{exec}_a(s)$ is defined by

- $s' \models e$
- $s(x) = s'(x)$ for all $x \in X$ that don’t occur in $e$.

Problem

Find $a_1, \ldots, a_n$ such that $\text{exec}_{a_n}(\text{exec}_{a_{n-1}}(\cdots \text{exec}_{a_2}(\text{exec}_{a_1}(I))\cdots)) \models G$?
Development of state-space search methods

A*

1968

1986

1988

1990

1992

1994

1996

1998

2000

partial-order reduction

symmetry reduction

BDDs

Symbolic Model-Checking

DNNF

SMTZ

Bounded Model-Checking

GRASP

SATPLAN

Planning as SAT

Chaff

Chaff

SMTZ

Bounded Model-Checking

GRASP

SATPLAN

Planning as SAT

Chaff
Symbolic Representations vs. Fwd and Bwd Search

symbolic data structures (BDD, DNNF, ...)

SAT

backward

forward

singleton backward

1. symbolic data structures
2. SAT
3. state-space search
4. others: partial-order planning [MR91] (until 1995)
Explicit State-Space Search

- The most basic search method for transition systems
- Very efficient for small state spaces (1 million states)
- Easy to implement
- Very well understood
- Pruning methods:
  - symmetry reduction [Sta91, ES96]
  - partial-order reduction [God91, Val91]
  - lower-bounds / heuristics, for informed search [HNR68]
State Representation

Each state represented explicitly ⇒ compact state representation important

- Boolean (0, 1) state variables represented by one bit
- Inter-variable dependencies enable further compaction:
  - \neg(at(A,L1) \land at(A,L2)) always true
  - automatic recognition of invariants [BF97, Rin98, Rin08]
  - \(n\) exclusive variables \(x_1, \ldots, x_n\) represented by \(1 + \lceil \log_2(n - 1) \rceil\) bits
Search Algorithms

- uninformed/blind search: depth-first, breadth-first, ...
- informed search: “best first” search (always expand best state so far)
- informed search: local search algorithms such as simulated annealing, tabu search and others [KGJV83, DS90, Glo89] (little used in planning)
- optimal algorithms: A* [HNR68], IDA* [Kor85]
Symmetry Reduction [Sta91, ES96]

Idea

1. Define an equivalence relation $\sim$ on the set of all states: $s_1 \sim s_2$ means that state $s_1$ is symmetric with $s_2$.
2. Only one state $s_C$ in each equivalence class $C$ needs to be considered.
3. If state $s \in C$ with $s \neq [s_C]$ is encountered, replace it with $s_C$.

Example

States $P(A) \land \neg P(B) \land P(C)$ and $\neg P(A) \land P(B) \land P(C)$ are symmetric because of the permutation $A \leftrightarrow B, B \leftrightarrow A, C \leftrightarrow C$. 

References
Symmetry Reduction
Example: 11 states, 3 equivalence classes
Symmetry Reduction
Example: 11 states, 3 equivalence classes
Symmetry Reduction
Example: 11 states, 3 equivalence classes
Symmetry Reduction
Example: 11 states, 3 equivalence classes
Symmetry Reduction
Example: 11 states, 3 equivalence classes
Symmetry Reduction

Example: 11 states, 3 equivalence classes
Idea [God91, Val91]

**Independent** actions unnecessary to consider in all orderings, e.g. both $A_1, A_2$ and $A_2, A_1$.

Example

Let there be lamps 1, 2, ..., $n$ which can be turned on. There are no other actions. One can restrict to plans in which lamps are turned on in the ascending order: switching lamp $n$ after lamp $m > n$ needless.$^a$

---

$^a$The same example is trivialized also by symmetry reduction!
The most basic heuristics widely used for non-optimal planning: 

- $h_{\text{max}}$ [BG01, McD96] best-known admissible heuristic 
- $h^+$ [BG01] still state-of-the-art 
- $h_{\text{relax}}$ [HN01] often more accurate, but performs like $h^+$
Definition of $h^{max}$, $h^+$ and $h^{relax}$

- Basic insight: estimate distances between possible state variable values, not states themselves.
- $g_s(l) = \begin{cases} 0 & \text{if } s \models l \\ \min_a \text{ with effect } p(1 + g_s(\text{prec}(a))) & \end{cases}$
- $h^+$ defines $g_s(L) = \sum_{l \in L} g_s(l)$ for sets $S$.
- $h^{max}$ defines $g_s(L) = \max_{l \in L} g_s(l)$ for sets $S$.
- $h^{relax}$ counts the number of actions in computation of $h^{max}$.
Computation of $h^{max}$

Tractor example

1. Tractor moves:
   - from 1 to 2: $T_{12} = \langle T_1, \{T_2, \neg T_1\} \rangle$
   - from 2 to 1: $T_{21} = \langle T_2, \{T_1, \neg T_2\} \rangle$
   - from 2 to 3: $T_{23} = \langle T_2, \{T_3, \neg T_2\} \rangle$
   - from 3 to 2: $T_{32} = \langle T_3, \{T_2, \neg T_3\} \rangle$

2. Tractor pushes A:
   - from 2 to 1: $A_{21} = \langle T_2 \land A_2, \{T_1, A_1, \neg T_2, \neg A_2\} \rangle$
   - from 3 to 2: $A_{32} = \langle T_3 \land A_3, \{T_2, A_2, \neg T_3, \neg A_3\} \rangle$

3. Tractor pushes B:
   - from 2 to 1: $B_{21} = \langle T_2 \land B_2, \{T_1, B_1, \neg T_2, \neg B_2\} \rangle$
   - from 3 to 2: $B_{32} = \langle T_3 \land B_3, \{T_2, B_2, \neg T_3, \neg B_3\} \rangle$
Computation of $h_{max}$

Tractor example

Apply $T_{12} = \langle T_1, \{T_2, \neg T_1\} \rangle$
Computation of $h_{\text{max}}$

Tractor example

<table>
<thead>
<tr>
<th>$t$</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
</tr>
<tr>
<td>4</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
</tr>
</tbody>
</table>

Apply $T23 = \langle T2, \{T3, \neg T2\} \rangle$
### Computation of $h_{max}$

#### Tractor example

<table>
<thead>
<tr>
<th>$t$</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
</tr>
<tr>
<td>4</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
</tr>
</tbody>
</table>

Apply $A_{32} = \langle T_3 \land A_3, \{T_2, A_2, \neg T_3, \neg A_3\} \rangle$
Computation of $h_{max}$

Tractor example

<table>
<thead>
<tr>
<th>$t$</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
</tr>
</tbody>
</table>

Apply $B_{32} = \langle T3 \land B3, \{T2, B2, \neg T3, \neg B3\} \rangle$
Computation of $h_{\text{max}}$

Tractor example

<table>
<thead>
<tr>
<th>$t$</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
</tr>
<tr>
<td>4</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
</tr>
</tbody>
</table>

Apply $A_{21} = \langle T2 \land A2, \{T1, A1, \neg T2, \neg A2\} \rangle$
**Computation of $h_{max}$**

**Tractor example**

<table>
<thead>
<tr>
<th>$t$</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
</tr>
<tr>
<td>4</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
</tr>
</tbody>
</table>

Apply $B_{21} = \langle T2 \land B2, \{T1, B1, \neg T2, \neg B2\} \rangle$
Distance of $A_1 \land B_1$ is 4.
Underestimates

Example

Estimate for \( \text{lamp1on} \land \text{lamp2on} \land \text{lamp3on} \) with

\[
\langle \top, \{\text{lamp1on}\} \rangle \\
\langle \top, \{\text{lamp2on}\} \rangle \\
\langle \top, \{\text{lamp3on}\} \rangle
\]

is 1. Actual shortest plan has length 3.

By definition, \( h^{max}(G_1 \land \cdots \land G_n) \) is the maximum of \( h^{max}(G_1), \ldots, h^{max}(G_n) \).

If goals are independent, the sum of the estimates is more accurate.
Computation of $h^+$

Tractor example

<table>
<thead>
<tr>
<th>$t$</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
</tr>
<tr>
<td>4</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
</tr>
<tr>
<td>5</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
</tr>
</tbody>
</table>

Apply $A21 = \langle T2 \land A2, \{T1, A1, \neg T2, \neg A2\}\rangle$.

$h^+(T2 \land A2)$ is $1+3$.

$h^+(A1)$ is $1+3+1 = 5$ ($h^{max}$ gives 4.)
Computation of $h^+$

Tractor example

<table>
<thead>
<tr>
<th>$t$</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
</tr>
<tr>
<td>4</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
</tr>
<tr>
<td>5</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
</tr>
</tbody>
</table>

$h^+$ of $A1 \land B1$ is $5 + 5 = 10$. 
Computation of $h^{relax}$

Motivation

<table>
<thead>
<tr>
<th>actions</th>
<th>estimate for $a \land b \land c$</th>
<th>actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle \top, {a, b, c} \rangle$</td>
<td>max \hspace{1cm} sum</td>
<td>1 \hspace{1cm} 3</td>
</tr>
<tr>
<td>$\langle \top, {a} \rangle, \langle \top, {b} \rangle, \langle \top, {c} \rangle$</td>
<td>1 \hspace{1cm} 3</td>
<td>3</td>
</tr>
</tbody>
</table>

- Better estimates with $h^{relax}$ (but: performance is similar to $h^+$).
- Application: directing search with preferred actions [Vid04, RH09]
Computation of $h_{relax}$

Estimate for $A1 \wedge B1$ with relaxed plans:

<table>
<thead>
<tr>
<th>$t$</th>
<th>relaxed plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
### Computation of $h_{relax}$

<table>
<thead>
<tr>
<th>$t$</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
<td>T</td>
<td>F</td>
<td>TF</td>
</tr>
<tr>
<td>4</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
</tr>
</tbody>
</table>

Estimate for $A1 \land B1$ with relaxed plans:

<table>
<thead>
<tr>
<th>$t$</th>
<th>relaxed plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>A21, B21</td>
</tr>
</tbody>
</table>
Computation of $h^{relax}$

<table>
<thead>
<tr>
<th>$t$</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
</tr>
<tr>
<td>4</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
</tr>
</tbody>
</table>

Estimate for $A1 \land B1$ with relaxed plans:

<table>
<thead>
<tr>
<th>$t$</th>
<th>relaxed plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>A21, B21</td>
</tr>
</tbody>
</table>
## Computation of $h^{relax}$

<table>
<thead>
<tr>
<th>$t$</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
</tr>
<tr>
<td>4</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
</tr>
</tbody>
</table>

Estimate for $A1 \land B1$ with relaxed plans:

<table>
<thead>
<tr>
<th>$t$</th>
<th>relaxed plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>A32, B32</td>
</tr>
<tr>
<td>3</td>
<td>A21, B21</td>
</tr>
</tbody>
</table>
### Computation of $h_{relax}$

<table>
<thead>
<tr>
<th>$t$</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
</tr>
<tr>
<td>4</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
</tr>
</tbody>
</table>

Estimate for $A1 \land B1$ with relaxed plans:

<table>
<thead>
<tr>
<th>$t$</th>
<th>relaxed plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>A32, B32</td>
</tr>
<tr>
<td>3</td>
<td>A21, B21</td>
</tr>
</tbody>
</table>
### Computation of $h_{relax}$

<table>
<thead>
<tr>
<th>$t$</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
</tr>
<tr>
<td>4</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
</tr>
</tbody>
</table>

Estimate for $A1 \land B1$ with relaxed plans:

<table>
<thead>
<tr>
<th>$t$</th>
<th>relaxed plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>T23</td>
</tr>
<tr>
<td>2</td>
<td>A32, B32</td>
</tr>
<tr>
<td>3</td>
<td>A21, B21</td>
</tr>
</tbody>
</table>
Computation of $h_{\text{relax}}$

<table>
<thead>
<tr>
<th>$t$</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
</tr>
<tr>
<td>4</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
</tr>
</tbody>
</table>

Estimate for $A1 \land B1$ with relaxed plans:

<table>
<thead>
<tr>
<th>$t$</th>
<th>relaxed plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>T23</td>
</tr>
<tr>
<td>2</td>
<td>A32, B32</td>
</tr>
<tr>
<td>3</td>
<td>A21, B21</td>
</tr>
</tbody>
</table>
# Computation of $h_{relax}$

<table>
<thead>
<tr>
<th>$t$</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
</tr>
<tr>
<td>4</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
</tr>
</tbody>
</table>

**Estimate for $A1 \land B1$ with relaxed plans:**

<table>
<thead>
<tr>
<th>$t$</th>
<th>relaxed plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T12</td>
</tr>
<tr>
<td>1</td>
<td>T23</td>
</tr>
<tr>
<td>2</td>
<td>A32, B32</td>
</tr>
<tr>
<td>3</td>
<td>A21, B21</td>
</tr>
</tbody>
</table>
Computation of $h^{relax}$

<table>
<thead>
<tr>
<th>$t$</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
<td>F</td>
<td>TF</td>
<td>TF</td>
</tr>
<tr>
<td>4</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
<td>TF</td>
</tr>
</tbody>
</table>

Estimate for $A1 \land B1$ with relaxed plans:

<table>
<thead>
<tr>
<th>$t$</th>
<th>relaxed plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T12</td>
</tr>
<tr>
<td>1</td>
<td>T23</td>
</tr>
<tr>
<td>2</td>
<td>A32, B32</td>
</tr>
<tr>
<td>3</td>
<td>A21, B21</td>
</tr>
</tbody>
</table>

estimate = number of actions in relaxed plan = 6
Comparison of the Heuristics

For the Tractor example:

- actions in the shortest plan: 8
- $h^{max}$ yields 4 (never overestimates).
- $h^+$ yields 10 (may under or overestimate).
- $h^{relax}$ yield 6 (may under or overestimate).

The sum-heuristic and the relaxed plan heuristic are used in practice for non-optimal planners.
Preferred Actions

- $h^+$ and $h^{relax}$ boosted with preferred/helpful actions.
- Preferred actions on the first level $t = 0$ in a relaxed plan.
- Several possibilities:
  - Always expand with a preferred action when possible [Vid04].
  - A tie-breaker when the heuristic values agree [RH09].
- Planners based on explicit state-space search use them: YAHSP, LAMA.
Performance of State-Space Search Planners
Planning Competition Problems

![Graph comparing the performance of various planners on STRIPS instances over time](image)

- **SS**: State-Space Search
- **HSP**: Hybrid Satisfiability Planning
- **FF**: Float Forward
- **LPG-td**: Learning Planning with Graphs - Transition-Directed
- **LAMA08**: Limited-Action Model A* 2008
- **YAHSP**: Yet Another Hybrid Satisfiability Planner

The graph plots the number of solved instances against the time in seconds for each planner. The x-axis represents time in seconds, ranging from 0 to 300, while the y-axis shows the number of solved instances, ranging from 0 to 1000.
Admissible heuristics are needed for finding optimal plans, e.g. with A* [HNR68]. Scalability much poorer.

**Pattern Databases [CS96, Ede00]**

Abstract away many/most state variables, and use the length/cost of the optimal solution to the remaining problem as an estimate.

**Generalized Abstraction (merge and shrink) [DFP09, HHH07]**

A generalization of pattern databases, allowing more complex aggregation of states (not just identification of ones agreeing on a subset of state variables.)

Landmark-cut [HD09] has been doing well with planning competition problems.
Planning with SAT

Background

- Proposed by Kautz and Selman [KS92].
- Idea as in Cook’s proof of NP-hardness of SAT [Coo71]: encode each step of a plan as a propositional formula.
- Intertranslatability of NP-complete problems $\Rightarrow$ reductions to many other problems possible.

Related solution methods

<table>
<thead>
<tr>
<th>Constraint satisfaction (CSP)</th>
<th>[vBC99, DK01]</th>
</tr>
</thead>
<tbody>
<tr>
<td>NM logic programs / answer-set programs</td>
<td>[DNK97]</td>
</tr>
</tbody>
</table>

Translations from SAT into other formalisms often simple. In terms of performance, SAT is usually the best choice.
Transition relations in propositional logic

State variables are
\( X = \{a, b, c\} \).

\[
\neg a \land b \land c \land \neg a' \land b' \land \neg c' \lor \\
\neg a \land b \land \neg c \land a' \land b' \land \neg c' \lor \\
\neg a \land \neg b \land c \land a' \land b' \land c' \lor \\
a \land b \land c \land a' \land b' \land \neg c'
\]

The corresponding matrix is

<table>
<thead>
<tr>
<th></th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>110</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
An action $j$ corresponds to the conjunction of the precondition $P_j @ t$ and

$$x_i (t + 1) \iff F_i(x_1 @ t, \ldots, x_n @ t)$$

for all $i \in \{1, \ldots, n\}$. Denote this by $E_j @ t$.

### Example (move-from-X-to-Y)

<table>
<thead>
<tr>
<th>precond</th>
<th>effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\neg atX @ t$</td>
<td>$(atX(t + 1) \iff \bot) \land (atY(t + 1) \iff \top) \land (atZ(t + 1) \iff atZ @ t) \land (atU(t + 1) \iff atU @ t)$</td>
</tr>
</tbody>
</table>

Choice between actions 1, \ldots, $m$ expressed by the formula

$$R @ t = E_1 @ t \lor \cdots \lor E_m @ t.$$
Finding a Plan with SAT

Let
- $I$ be a formula expressing the initial state, and
- $G$ be a formula expressing the goal states.

Then a plan of length $T$ exists iff

$$I_{@0} \land \bigwedge_{t=0}^{T-1} R_{@t} \land G_T$$

is satisfiable.

Remark

Most SAT solvers require formulas to be in CNF. There are efficient transformations to achieve this [Tse62, JS05, MV07].
Parallel Plans: Motivation

- Don’t represent all intermediate states of a sequential plan.
- Ignore relative ordering of consecutive actions.
- Reduced number of explicitly represented states $\Rightarrow$ smaller formulas
Parallel plans (∀-step plans)
Kautz and Selman 1996

Allow actions $a_1 = \langle p_1, e_1 \rangle$ and $a_2 = \langle p_2, e_2 \rangle$ in parallel whenever they don’t interfere, i.e.

- both $p_1 \cup p_2$ and $e_1 \cup e_2$ are consistent, and
- both $e_1 \cup p_2$ and $e_2 \cup p_1$ are consistent.

**Theorem**

If $a_1 = \langle p_1, e_1 \rangle$ and $a_2 = \langle p_1, e_1 \rangle$ don’t interfere and $s$ is a state such that $s \models p_1$ and $s \models p_2$, then

$$\text{exec}_{a_1}(\text{exec}_{a_2}(s)) = \text{exec}_{a_2}(\text{exec}_{a_1}(s)).$$
∀-step plans: encoding

Define $R^\forall@t$ as the conjunction of

$$x@t(t + 1) \leftrightarrow ((x@t \land \lnot a_1@t \land \cdots \land \lnot a_k@t) \lor a'_1@t \lor \cdots \lor a'_k@t)$$

for all $x \in X$, where $a_1, \ldots, a_k$ are all actions making $x$ false, and $a'_1, \ldots, a'_k$ are all actions making $x$ true, and

$$a@t \rightarrow l@t \text{ for all } l \text{ in the precondition of } a,$$

and

$$\lnot(a@t \land a'@t) \text{ for all } a \text{ and } a' \text{ that interfere.}$$

This encoding is quadratic due to the interference clauses.
∀-step plans: linear encoding
Rintanen et al. 2006 [RHN06]

Action $a$ with effect $l$ disables all actions with precondition $\bar{l}$, except $a$ itself. This is done in two parts: disable actions with higher index, disable actions with lower index.

This is needed for every literal.
Allow actions \( \{a_1, \ldots, a_n\} \) in parallel if they can be executed in at least one order.

- \( \bigcup_{i=1}^{n} p_i \) is consistent.
- \( \bigcup_{i=1}^{n} e_i \) is consistent.
- There is a total ordering \( a_1, \ldots, a_n \) such that \( e_i \cup p_j \) is consistent whenever \( i \leq j \): disabling an action earlier in the ordering is allowed.

Several compact encodings exist [RHN06]. Fewer time steps are needed than with \( \forall \)-step plans. Sometimes only half as many.
Choose an arbitrary fixed ordering of all actions $a_1, \ldots, a_n$.

Action $a$ with effect $l$ disables all later actions with precondition $\bar{l}$.

This is needed for every literal.
Define a disabling graph with actions as nodes and with an arc from \( a_1 \) to \( a_2 \) (\( a_1 \) disables \( a_2 \)) if \( p_1 \cup p_2 \) and \( e_1 \cup e_2 \) are consistent and \( e_1 \cup p_2 \) is inconsistent.

The test for valid execution orderings can be limited to strongly connected components (SCC) of the disabling graph.

In many structured problems all SCCs are singleton sets. \( \implies \) No tests for validity of orderings needed during SAT solving.
Summary of Notions of Plans

<table>
<thead>
<tr>
<th>plan type</th>
<th>reference</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>sequential</td>
<td>[KS92]</td>
<td>one action per time point</td>
</tr>
<tr>
<td>$\forall$-parallel</td>
<td>[BF97, KS96]</td>
<td>parallel actions independent</td>
</tr>
<tr>
<td>$\exists$-parallel</td>
<td>[DNK97, RHN06]</td>
<td>executable in at least one order</td>
</tr>
</tbody>
</table>

The last two expressible in terms of the relation disables restricted to applied actions:

- $\forall$-parallel plans: the disables relation is empty.
- $\exists$-parallel plans: the disables relation is acyclic.
The planning problem is reduced to the satisfiability tests for

\[
\begin{align*}
\Phi_0 &= I@0 \land G@0 \\
\Phi_1 &= I@0 \land R@0 \land G@1 \\
\Phi_2 &= I@0 \land R@0 \land R@1 \land G@2 \\
\Phi_3 &= I@0 \land R@0 \land R@1 \land R@2 \land G@3 \\
\vdots \\
\Phi_u &= I@0 \land R@0 \land R@1 \land \cdots R@(u - 1) \land G@u
\end{align*}
\]

where \( u \) is the maximum possible plan length.

Q: How to schedule these satisfiability tests?
## Search through Horizon Lengths

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Reference</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>sequential</td>
<td>[KS92, KS96]</td>
<td>slow, guarantees min. horizon</td>
</tr>
<tr>
<td>binary search</td>
<td>[SS07]</td>
<td>prerequisite: length UB</td>
</tr>
<tr>
<td>$n$ processes</td>
<td>[Rin04b, Zar04]</td>
<td>fast, more memory needed</td>
</tr>
<tr>
<td>geometric</td>
<td>[Rin04b]</td>
<td>fast, more memory needed</td>
</tr>
</tbody>
</table>

- **sequential**: first test $\Phi_0$, then $\Phi_1$, then $\Phi_2$, ...  
  - This is breadth-first search / iterative deepening.  
  - Guarantees shortest horizon length, but is slow.  

- **parallel strategies**: solve several horizon lengths simultaneously  
  - depth-first flavor  
  - usually much faster  
  - no guarantee of minimal horizon length
Some runtime profiles

![Runtime Profile](image-url)
Some runtime profiles
Some runtime profiles
Some runtime profiles

![Histogram of evaluation times for schedule51](image)

- **Evaluation times:** schedule51
- **Y-axis:** time in secs
- **X-axis:** time points
Some runtime profiles

Evaluation times: blocks22

- Time in secs
- Time points

0 10 20 30 40 50 60 70 80 90 100

0 5 10 15 20 25 30 35 40
Some runtime profiles

Evaluation times: depot15

- Time in secs vs. time points
- Peaks at approximately time point 20 with high evaluation times
Finding a plan for blocks22 with Algorithm B
Finding a plan for blocks22 with Algorithm B
Finding a plan for blocks22 with Algorithm B
Finding a plan for blocks22 with Algorithm B
Finding a plan for blocks22 with Algorithm B
Finding a plan for blocks22 with Algorithm B
Finding a plan for blocks22 with Algorithm B
Finding a plan for blocks22 with Algorithm B

Geometric Evaluation
Solving the SAT Problem

SAT problems obtained from planning are solved by

- **generic SAT solvers**
  - Mostly based on **Conflict-Driven Clause Learning (CDCL)** [MMZ+01].
  - Extremely good on hard combinatorial planning problems.
  - Not designed for solving the extremely large but “easy” formulas (arising in some types of benchmark problems).

- **specialized SAT solvers** [Rin10b, Rin10a]
  - Replace standard CDCL heuristics with planning-specific ones.
  - For certain problem classes substantial improvement
  - New research topic: lots of unexploited potential
Problem solved almost without search:

- Formulas for lengths 1 to 4 shown unsatisfiable without any search.
- Formula for plan length 5 is satisfiable: 3 nodes in the search tree.
- Plans have 5 to 7 operators, optimal plan has 5.
Solving the SAT Problem

Example

1. State variable values inferred from initial values and goals.
2. Branch: \( \neg \text{clear(b)} \)\(^1\).
3. Branch: clear(a)\(^3\).
4. Plan found:

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 & 5 \\
\end{array}
\]

\[
\begin{array}{ccccccc}
\text{clear(a)} & F & F & F & F & F & F \\
\text{clear(b)} & F & F & F & F & F & F \\
\text{clear(c)} & T & T & F & T & F & T \\
\text{clear(d)} & F & T & F & F & F & F \\
\text{clear(e)} & T & T & F & F & F & F \\
\text{on(a,b)} & F & F & F & F & F & F \\
\text{on(a,c)} & F & F & F & F & F & F \\
\text{on(a,e)} & F & F & F & F & F & F \\
\text{on(b,a)} & T & T & F & F & F & F \\
\text{on(b,c)} & F & F & T & T & T & T \\
\text{on(b,d)} & F & F & F & F & F & F \\
\text{on(b,e)} & F & F & F & F & F & F \\
\text{on(c,a)} & F & F & F & F & F & F \\
\text{on(c,b)} & T & F & T & T & T & T \\
\text{on(c,d)} & F & F & T & T & T & T \\
\text{on(c,e)} & F & F & F & F & F & F \\
\text{on(d,a)} & F & F & F & F & F & F \\
\text{on(d,b)} & F & F & F & F & F & F \\
\text{on(d,c)} & F & F & F & F & F & F \\
\text{on(d,e)} & F & F & T & T & T & T \\
\text{on(e,a)} & F & F & F & F & F & F \\
\text{on(e,b)} & F & F & F & F & F & F \\
\text{on(e,c)} & F & F & F & F & F & F \\
\text{on(e,d)} & T & F & T & T & T & T \\
\text{ontable(a)} & T & T & T & F & T & T \\
\text{ontable(b)} & F & F & F & F & F & F \\
\text{ontable(c)} & F & F & F & F & F & F \\
\text{ontable(d)} & T & F & T & T & T & T \\
\text{ontable(e)} & F & T & T & T & T & T \\
\end{array}
\]
Solving the SAT Problem

Example

1. State variable values inferred from initial values and goals.
2. Branch: $\neg$clear(b)$^1$.
3. Branch: clear(a)$^3$.
4. Plan found:

\begin{align*}
\text{clear(a)} & \quad F \quad F \quad F \quad T \\
\text{clear(b)} & \quad F \quad F \quad T \\
\text{clear(c)} & \quad T \quad T \quad F \\
\text{clear(d)} & \quad F \quad T \quad F \\
\text{clear(e)} & \quad T \quad T \quad F \\
\text{on(a,b)} & \quad F \quad F \quad T \\
\text{on(a,c)} & \quad F \quad F \quad F \\
\text{on(a,d)} & \quad F \quad F \quad F \\
\text{on(a,e)} & \quad F \quad F \quad F \\
\text{on(b,a)} & \quad T \quad T \quad F \\
\text{on(b,c)} & \quad F \quad F \quad T \\
\text{on(b,d)} & \quad F \quad F \quad F \\
\text{on(b,e)} & \quad F \quad F \quad F \\
\text{on(c,a)} & \quad F \quad F \quad F \\
\text{on(c,b)} & \quad F \quad T \\
\text{on(c,d)} & \quad F \quad F \quad T \\
\text{on(c,e)} & \quad F \quad F \quad F \\
\text{on(d,a)} & \quad F \quad F \quad F \\
\text{on(d,b)} & \quad F \quad F \quad F \\
\text{on(d,c)} & \quad F \quad F \quad F \\
\text{on(d,e)} & \quad F \quad F \quad F \\
\text{on(e,a)} & \quad F \quad F \quad F \\
\text{on(e,b)} & \quad F \quad F \quad F \\
\text{on(e,c)} & \quad F \quad F \quad F \\
\text{on(e,d)} & \quad T \quad F \\
\text{ontable(a)} & \quad T \quad T \quad F \\
\text{ontable(b)} & \quad F \quad F \quad F \\
\text{ontable(c)} & \quad F \quad F \quad F \\
\text{ontable(d)} & \quad T \quad T \quad F \\
\text{ontable(e)} & \quad F \quad T \quad T
\end{align*}

\begin{align*}
\text{fromtable(a,b)} & \quad F \quad F \quad F \\
\text{fromtable(b,c)} & \quad F \quad F \quad T \\
\text{fromtable(c,d)} & \quad F \quad T \\
\text{fromtable(d,e)} & \quad T \quad F \\
\text{totable(b,a)} & \quad F \quad T \\
\text{totable(c,b)} & \quad F \quad T \\
\text{totable(e,d)} & \quad T \quad F
\end{align*}
Solving the SAT Problem

Example

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear(a)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>clear(b)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>clear(c)</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>clear(d)</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>clear(e)</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>on(a,b)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>on(a,c)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>on(a,d)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>on(b,a)</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>on(b,c)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>on(b,d)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>on(b,e)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>on(c,a)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>on(c,b)</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>on(c,d)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>on(c,e)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>on(d,a)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>on(d,b)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>on(d,c)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>on(d,e)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>on(e,a)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>on(e,b)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>on(e,c)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>on(e,d)</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>ontable(a)</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>ontable(b)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>ontable(c)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>ontable(d)</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>ontable(e)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

1. State variable values inferred from initial values and goals.
2. Branch: $\neg$clear(b)\(^1\).
3. Branch: clear(a)\(^3\).
4. Plan found:

```
0 1 2 3 4
fromtable(a,b) F F F F T
fromtable(b,c) F F F T F
fromtable(c,d) F F T F F
fromtable(d,e) F T F F F
totable(b,a) F F F F F
totable(c,b) F F F F F
totable(e,d) T F F F F
```
Solving the SAT Problem

Example

<table>
<thead>
<tr>
<th>0 1 2 3 4 5</th>
<th>0 1 2 3 4 5</th>
<th>0 1 2 3 4 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear(a)</td>
<td>F F</td>
<td>F F</td>
</tr>
<tr>
<td>clear(b)</td>
<td>F F</td>
<td>F F</td>
</tr>
<tr>
<td>clear(c)</td>
<td>T T</td>
<td>F F</td>
</tr>
<tr>
<td>clear(d)</td>
<td>F T T T F F</td>
<td>F T T F F F</td>
</tr>
<tr>
<td>clear(e)</td>
<td>T F T F F F</td>
<td>T F T F F F</td>
</tr>
<tr>
<td>on(a,b)</td>
<td>F T</td>
<td>F F</td>
</tr>
<tr>
<td>on(a,c)</td>
<td>F F F F F F</td>
<td>F F F F F F</td>
</tr>
<tr>
<td>on(a,d)</td>
<td>F F F F F F</td>
<td>F F F F F F</td>
</tr>
<tr>
<td>on(a,e)</td>
<td>F F F F F F</td>
<td>F F F F F F</td>
</tr>
<tr>
<td>on(b,a)</td>
<td>T T</td>
<td>T T</td>
</tr>
<tr>
<td>on(b,c)</td>
<td>F T T T T T</td>
<td>F T T T T T</td>
</tr>
<tr>
<td>on(b,d)</td>
<td>F F F F F F</td>
<td>F F F F F F</td>
</tr>
<tr>
<td>on(b,e)</td>
<td>F F F F F F</td>
<td>F F F F F F</td>
</tr>
<tr>
<td>on(c,a)</td>
<td>F F F F F F</td>
<td>F F F F F F</td>
</tr>
<tr>
<td>on(c,b)</td>
<td>T F F F</td>
<td>T F F F</td>
</tr>
<tr>
<td>on(c,d)</td>
<td>F F T T T T</td>
<td>F F T T T T</td>
</tr>
<tr>
<td>on(c,e)</td>
<td>F F F F F F</td>
<td>F F F F F F</td>
</tr>
<tr>
<td>on(d,a)</td>
<td>F F F F F F</td>
<td>F F F F F F</td>
</tr>
<tr>
<td>on(d,b)</td>
<td>F F F F F F</td>
<td>F F F F F F</td>
</tr>
<tr>
<td>on(d,c)</td>
<td>F F F F F F</td>
<td>F F F F F F</td>
</tr>
<tr>
<td>on(d,e)</td>
<td>F T T T T T</td>
<td>F T T T T T</td>
</tr>
<tr>
<td>on(e,a)</td>
<td>F F F F F F</td>
<td>F F F F F F</td>
</tr>
<tr>
<td>on(e,b)</td>
<td>F F F F F F</td>
<td>F F F F F F</td>
</tr>
<tr>
<td>on(e,c)</td>
<td>F F F F F F</td>
<td>F F F F F F</td>
</tr>
<tr>
<td>on(e,d)</td>
<td>F F F F F F</td>
<td>F F F F F F</td>
</tr>
</tbody>
</table>

1. State variable values inferred from initial values and goals.
2. Branch: \( \neg \text{clear(b)} \).
3. Branch: \text{clear(a)}.
4. Plan found:

<table>
<thead>
<tr>
<th>0 1 2 3 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>fromtable(a,b) F F F F T</td>
</tr>
<tr>
<td>fromtable(b,c) F F F T F</td>
</tr>
<tr>
<td>fromtable(c,d) F F T F F</td>
</tr>
<tr>
<td>fromtable(d,e) F T F F F</td>
</tr>
<tr>
<td>totable(b,a) F F T F F</td>
</tr>
<tr>
<td>totable(c,b) F T F F F</td>
</tr>
<tr>
<td>totable(e,d) T F F F F</td>
</tr>
</tbody>
</table>
Performance of SAT-Based Planners
Planning Competition Problems 1998-2008

![Graph showing the performance of various planners on STRIPS instances](image)

- SS
- HSP
- FF
- LPG-td
- LAMA08
- YAHSP

The graph illustrates the number of solved instances over time for different planners on STRIPS instances. The x-axis represents time in seconds, ranging from 0 to 300, and the y-axis represents the number of solved instances, ranging from 0 to 1000.
Performance of SAT-Based Planners
Planning Competition Problems 1998-2008

![Graph showing the performance of different planners over time]

- SS
- HSP
- FF
- LPG-td
- LAMA08
- YAHSP
- SATPLAN
- M
- Mp
Performance of SAT-Based Planners
Planning Competition Problems 1998-2011 (revised)

all domains 1998-2011

- SATPLAN
- M
- Mp
- MpX
- LAMA08
- LAMA11
- FF
- FF-2

number of solved instances vs. time in seconds

- SATPLAN
- M
- Mp
- MpX
- LAMA08
- LAMA11
- FF
- FF-2
MathSAT [BBC+05] and other SAT modulo Theories (SMT) solvers extend SAT with numerical variables and equalities and inequalities. Applications include:

- timed systems [ACKS02], temporal planning
- hybrid systems [GPB05, ABCS05], temporal planning + continuous change
Symbolic Search Methods

Motivation

- **logical formulas as a data structure** for sets, relations
- Planning (model-checking, diagnosis, ...) algorithms in terms of set & relational operations.
- Algorithms that can handle **very large** state sets efficiently, bypassing inherent limitations of explicit state-space search.
- **Complementary** to explicit (enumerative) representations of state sets: strengths in different types of problems.
Transition relations in propositional logic

State variables are
\( X = \{a, b, c\} \).

\[
\begin{align*}
(\neg a \land b \land c \land \neg a' \land b' \land \neg c') \lor \\
(\neg a \land b \land \neg c \land a' \land b' \land \neg c') \lor \\
(\neg a \land \neg b \land c \land a' \land b' \land c') \lor \\
(a \land b \land c \land a' \land b' \land \neg c')
\end{align*}
\]

The corresponding matrix is

<table>
<thead>
<tr>
<th></th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>110</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
The image of a set $T$ of states w.r.t. action $a$ is

$$\text{img}_a(T) = \{ s' \in S | s \in T, sas' \}.$$ 

The pre-image of a set $T$ of states w.r.t. action $a$ is

$$\text{preimg}_a(T) = \{ s \in S | s' \in T, sas' \}.$$ 

These operations reduce to the relational join and projection operations with a logic-representation of sets (unary relations) and binary relations.
Finding Plans with a Symbolic Algorithm

Computation of all reachable states

\[ S_0 = \{I\} \]
\[ S_{i+1} = S_i \cup \bigcup_{x \in X} \text{img}_x(S_i) \]

If \( S_i = S_{i+1} \), then \( S_j = S_i \) for all \( j \geq i \), and the computation can be terminated.

- \( S_i, i \geq 0 \) is the set of states with distance \( \leq i \) from the initial state.
- \( S_i \setminus S_{i-1}, i \geq 1 \) is the set of states with distance \( i \).
- If \( G \cap S_i \) for some \( i \geq 0 \), then there is a plan.

Action sequence recovered from sets \( S_i \) by a sequence of backward-chaining steps.
Use in Connection with Heuristic Search Algorithms

Symbolic (BDD) versions of heuristic algorithms in the state-space search context:

- SetA* [JVB08]
- BDDA* [ER98]
- ADDA* [HZF02]
BDDs and other normal forms standard representation in planning with partial observability [BCRT01, Rin05]. Also, probabilistic planning [HSAHB99] with value functions represented as Algebraic Decision Diagrams (ADD) [FMY97, BFG+97].

A belief state is a set of possible current states.

These sets are often very large, best represented as formulas.
Significance of Symbolic Representations

- Much more powerful framework than SAT or explicit state-space search.
- Unlike other methods, allows exhaustive generation of reachable states.
- Problem 1: e.g. with BDDs, size of transition relation may explode.
- Problem 2: e.g. with BDDs, size of sets $S_i$ may explode.
- Important research topic: symbolic search with less restrictive normal forms than BDD.
Images as Relational Operations

\[
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
 s_0 \\
 s_2 \\
 s_1 \\
 s_0 \\
 s_2 \\
 s_2 \\
\end{array}
\end{array}
\end{array}
\times
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
 s_0 \\
 s_2 \\
 s_0 \\
 s_2 \\
 s_1 \\
 s_0 \\
\end{array}
\end{array}
\end{array}
= 
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
 s_0 \\
 s_1 \\
 s_0 \\
 s_2 \\
 s_1 \\
 s_0 \\
\end{array}
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
 s_0 \\
 s_0 \\
 s_2 \\
 s_2 \\
 s_0 \\
 s_0 \\
\end{array}
\end{array}
\end{array}
\]
Images as Relational Operations

<table>
<thead>
<tr>
<th>s₀</th>
<th>s₁</th>
<th>s₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>s₀</td>
<td>s₁</td>
<td>s₂</td>
</tr>
<tr>
<td>s₁</td>
<td>s₀</td>
<td>s₂</td>
</tr>
<tr>
<td>s₂</td>
<td>s₀</td>
<td>s₁</td>
</tr>
</tbody>
</table>

×

<table>
<thead>
<tr>
<th>s₀₀</th>
<th>s₁₀</th>
<th>s₂₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>s₀₀</td>
<td>s₁₀</td>
<td>s₂₁</td>
</tr>
<tr>
<td>s₁₀</td>
<td>s₀₀</td>
<td>s₂₁</td>
</tr>
<tr>
<td>s₂₁</td>
<td>s₀₀</td>
<td>s₁₀</td>
</tr>
</tbody>
</table>

= 

| s₀s₁₀₀₀₁ |
| s₀s₂₀₀₁₀ |
| s₁s₀₀₁₀₀ |
| s₁s₂₀₁₁₀ |
| s₂s₀₁₀₀₀ |

×

| s₀s₁₀₀₀₁ |
| s₀s₂₀₀₁₀ |
| s₁s₀₀₁₀₀ |
| s₁s₂₀₁₁₀ |
| s₂s₀₁₀₀₀ |

= 

| s₀s₁₀₀₀₁ |
| s₀s₂₀₀₁₀ |
| s₁s₀₀₁₀₀ |
| s₁s₂₀₁₁₀ |
| s₂s₀₁₀₀₀ |
Images as Relational Operations

\[
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
 s_0 00 \\
 s_2 10
\end{array}
\end{array}
\times
\begin{array}{c}
\begin{array}{c}
 s_0 s_1 00 01 \\
 s_0 s_2 00 10 \\
 s_1 s_0 01 00 \\
 s_1 s_2 01 10 \\
 s_2 s_0 10 00
\end{array}
\end{array}
\end{array}
= \begin{array}{c}
\begin{array}{c}
 x_0 x_1 \\
 00 \\
 01 \\
 10 \\
 11
\end{array}
\end{array}
\times
\begin{array}{c}
\begin{array}{c}
 x_0 x_1' x_0 x_1 \\
 0000 \\
 0001 \\
 0010 \\
 0011 \\
 0100 \\
 0101 \\
 0110 \\
 0111 \\
 1000 \\
 1001 \\
 1010 \\
 1011 \\
 1100 \\
 1101 \\
 1110 \\
 1111
\end{array}
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
 x_0 x_1 x_0 x_1 \\
 0001 \\
 0010 \\
 0110 \\
 1000 \\
 1001 \\
 1010 \\
 1011 \\
 1100 \\
 1101 \\
 1110 \\
 1111
\end{array}
\end{array}
= \begin{array}{c}
\begin{array}{c}
 x_0 x_1 x_0 x_1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 1
\end{array}
\end{array}
\]
## Representation of Sets as Formulas

<table>
<thead>
<tr>
<th>state sets</th>
<th>formulas over $X$</th>
</tr>
</thead>
<tbody>
<tr>
<td>those $\frac{2^{</td>
<td>X</td>
</tr>
<tr>
<td>$\overline{E}$</td>
<td>$\neg E$</td>
</tr>
<tr>
<td>$E \cup F$</td>
<td>$E \lor F$</td>
</tr>
<tr>
<td>$E \cap F$</td>
<td>$E \land F$</td>
</tr>
<tr>
<td>$E \setminus F$</td>
<td>$E \land \neg F$</td>
</tr>
<tr>
<td>the empty set $\emptyset$</td>
<td>$\bot$ (constant false)</td>
</tr>
<tr>
<td>the universal set $\top$</td>
<td>$\top$ (constant true)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>question about sets</th>
<th>question about formulas</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E \subseteq F$?</td>
<td>$E \models F$?</td>
</tr>
<tr>
<td>$E \subset F$?</td>
<td>$E \models F$ and $F \not\models E$?</td>
</tr>
<tr>
<td>$E = F$?</td>
<td>$E \models F$ and $F \models E$?</td>
</tr>
</tbody>
</table>
Sets (of states) as formulas

Formulas over $X$ represent sets

$$a \lor b \text{ over } X = \{a, b, c\}$$
represents the set $\{010, 011, 100, 101, 110, 111\}$.

Formulas over $X \cup X'$ represent binary relations

$$a \land a' \land (b \leftrightarrow b') \text{ over } X \cup X' \text{ where } X = \{a, b\}, \ X' = \{a', b'\}$$
represents the binary relation $\{(a, a'), (11, 11)\}$.
Valuations $1010$ and $1111$ of $X \cup X'$ can be viewed respectively as pairs of valuations $\langle a b, a'b' \rangle$ and $\langle 10, 11 \rangle$ of $X$. 
<table>
<thead>
<tr>
<th>relation operation</th>
<th>logical operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>projection</td>
<td>abstraction</td>
</tr>
<tr>
<td>join</td>
<td>conjunction</td>
</tr>
</tbody>
</table>
Normal Forms

<table>
<thead>
<tr>
<th>normal form</th>
<th>reference</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNF Negation Normal Form</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNF Disjunctive Normal Form</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNF Conjunctive Normal Form</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BDD Binary Decision Diagram</td>
<td>[Bry92]</td>
<td>most popular</td>
</tr>
<tr>
<td>DNNF Decomposable NNF</td>
<td>[Dar01]</td>
<td>more compact</td>
</tr>
</tbody>
</table>

Darwiche’s terminology: knowledge compilation languages [DM02]

Trade-off

- more compact $\leftrightarrow$ less efficient operations
- But, “more efficient” is in the size of a correspondingly inflated formula. (Also more efficient in terms of wall clock?) BDD-SAT is $O(1)$, but e.g. translation into BDDs is (usually) far less efficient than testing SAT directly.
Complexity of Operations

Operations offered e.g. by BDD packages:

<table>
<thead>
<tr>
<th></th>
<th>∨</th>
<th>∧</th>
<th>¬</th>
<th>$\phi \in$ TAUT?</th>
<th>$\phi \in$ SAT?</th>
<th>$\phi \equiv \phi'$?</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNF</td>
<td>poly</td>
<td>poly</td>
<td>poly</td>
<td>co-NP-hard</td>
<td>NP-hard</td>
<td>co-NP-hard</td>
</tr>
<tr>
<td>DNF</td>
<td>poly</td>
<td>exp</td>
<td>exp</td>
<td>co-NP-hard</td>
<td>in P</td>
<td>co-NP-hard</td>
</tr>
<tr>
<td>CNF</td>
<td>exp</td>
<td>poly</td>
<td>exp</td>
<td>in P</td>
<td>NP-hard</td>
<td>co-NP-hard</td>
</tr>
<tr>
<td>BDD</td>
<td>exp</td>
<td>exp</td>
<td>poly</td>
<td>in P</td>
<td>in P</td>
<td>in P</td>
</tr>
</tbody>
</table>

Remark

*For BDDs one $\lor / \land$ is polynomial time/size (size is doubled) but repeated $\lor / \land$ lead to exponential size.*
Existential and Universal Abstraction

**Definition**

**Existential abstraction** of a formula $\phi$ with respect to $x \in X$:

$$\exists x.\phi = \phi[\top/x] \lor \phi[\bot/x].$$

Universal abstraction is defined analogously by using conjunction instead of disjunction.

**Definition**

**Universal abstraction** of a formula $\phi$ with respect to $x \in X$:

$$\forall x.\phi = \phi[\top/x] \land \phi[\bot/x].$$
∃-Abstraction

Example

\[ \exists b. ((a \to b) \land (b \to c)) \]
\[ = ((a \to T) \land (T \to c)) \lor ((a \to \bot) \land (\bot \to c)) \]
\[ \equiv c \lor \neg a \]
\[ \equiv a \to c \]

\[ \exists a b. (a \lor b) = \exists b. (T \lor b) \lor (\bot \lor b) \]
\[ = ((T \lor T) \lor (\bot \lor T)) \lor ((T \lor \bot) \lor (\bot \lor \bot)) \]
\[ \equiv (T \lor T) \lor (T \lor \bot) \equiv T \]
∀c and ∃c correspond to combining lines with the same valuation for variables other than c.

Example

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>a ∨ (b ∧ c)</th>
<th>∀c.(a ∨ (b ∧ c))</th>
<th>a</th>
<th>b</th>
<th>∃c.(a ∨ (b ∧ c))</th>
<th>a</th>
<th>b</th>
<th>∀c.(a ∨ (b ∧ c))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Let $X$ be the set of all state variables. An action $a$ corresponds to the conjunction of the precondition $P_j$ and

$$
x' \leftrightarrow F_i(X)
$$

for all $x \in X$. Denote this by $\tau_X(a)$.

**Example (move-from-A-to-B)**

$$atA \land (atA' \leftrightarrow \bot) \land (atB' \leftrightarrow \top) \land (atC' \leftrightarrow atC) \land (atD' \leftrightarrow atD)$$

This is exactly the same as in the SAT case, except that we have $x$ and $x'$ instead of $x@t$ and $x@(t + 1)$.
Computation of Successor States

Let
- \( X = \{x_1, \ldots, x_n\} \),
- \( X' = \{x'_1, \ldots, x'_n\} \),
- \( \phi \) be a formula over \( X \) that represents a set \( T \) of states.

Image Operation

The image \( \{s' \in S | s \in T, sas'\} \) of \( T \) with respect to \( a \) is

\[
\text{img}_a(\phi) = (\exists X. (\phi \land \tau_X(a)))[X/X'].
\]

The renaming is necessary to obtain a formula over \( X \).
Computation of Predecessor States

Let
- \( X = \{x_1, \ldots, x_n\} \),
- \( X' = \{x'_1, \ldots, x'_n\} \),
- \( \phi \) be a formula over \( X \) that represents a set \( T \) of states.

Preimage Operation

The pre-image \( \{s \in S | s' \in T, sas'\} \) of \( T \) with respect to \( a \) is

\[
\text{preimg}_a(\phi) = (\exists X'. (\phi[X'/X] \land \tau_X(a))).
\]

The renaming of \( \phi \) is necessary so that we can start with a formula over \( X \).
Engineering Efficient Planners

- Gap between Theory and Practice large: engineering details of implementation critical for performance in current planners.
- Few of the most efficient planners use textbook methods.
- Explanations for the observed differences between planners lacking: this is more art than science.
Algorithm Portfolios

- Algorithm portfolio = combination of two or more algorithms
- Useful if there is no single “strongest” algorithm.
Composition methods:

- **selection** = choose one, for the instance in question
- **parallel** composition = run components in parallel
- **sequential** composition = run consecutively, according to a schedule

Examples: BLACKBOX [KS99], FF [HN01], LPG [GS02] (all use sequential composition)
Algorithm Portfolios
An Illustration of Portfolios

![Graph showing STRIPS instances solved over time for different algorithms.](image)

- **FF** = FF-1 followed by FF-2
- **LPG-td** = LPGT-td-1 followed by FF-2
Evaluation of Planners

Evaluation of planning systems is based on

- Hand-crafted problems (from the planning competitions)
  - This is the most popular option.
  - Problems with (at least moderately) different structure.
  - Real-world relevance mostly low.
  - Instance generation uncontrolled: not known if easy or difficult.
  - Many have a similar structure: objects moving in a network.

- Benchmark sets obtained by translation from other problems
  - graph-theoretic problems: cliques, colorability, ... [PMB11]

- Instances sampled from all instances [Byl96].
  - Easy to control problem hardness.
  - No direct real-world relevance (but: core of any “hard” problem)
Sampling from the Set of All Instances
[Byl96, Rin04c]

- **Generation:**
  1. Fix number $N$ of state variables, number $M$ of actions.
  2. For each action, choose preconditions and effects randomly.
- Has a **phase transition** from unsolvable to solvable, similarly to SAT [MSL92] and connectivity of random graphs [Bol85].
- Exhibits an **easy-hard-easy** pattern, for a fixed $N$ and an increasing $M$, analogously to SAT [MSL92].
- Hard instances roughly at the 50 per cent solvability point.
- Hardest instances are **very hard**: 20 state variables too difficult for many planners, as their heuristics don’t help.
Sampling from the Set of All Instances
Experiments with planners

Model A: Distribution of runtimes with SAT

runtime in seconds
ratio operators / state variables

100
10
1
0.1
0.01
0.001

1.5 2 2.5 3 3.5 4 4.5 5 5.5 6
Model A: Distribution of runtimes with FF

- **x-axis**: Ratio of operators to state variables
- **y-axis**: Runtime in seconds

The graph illustrates the distribution of runtimes for different ratios of operators to state variables, with the x-axis ranging from 1.5 to 6 and the y-axis ranging from 0.01 to 100 seconds.
Sampling from the Set of All Instances
Experiments with planners

Model A: Distribution of runtimes with HSP

runtime in seconds
ratio operators / state variables

Model A: Distribution of runtimes with HSP
Gilles Audemard, Marco Bozzano, Alessandro Cimatti, and Roberto Sebastiani.
Verifying industrial hybrid systems with MathSAT.

Gilles Audemard, Alessandro Cimatti, Artur Korniłowicz, and Roberto Sebastiani.
Bounded model checking for timed systems.

Marco Bozzano, Roberto Bruttomesso, Alessandro Cimatti, Tommi Junttila, Peter van Rossum, Stephan Schulz, and Roberto Sebastiani.
The MathSAT 3 system.

Piergiorgio Bertoli, Alessandro Cimatti, Marco Roveri, and Paolo Traverso.
Planning in nondeterministic domains under partial observability via symbolic model checking.

Avrim L. Blum and Merrick L. Furst.
Fast planning through planning graph analysis.
Algebraic decision diagrams and their applications.

Blai Bonet and Héctor Geffner.
Planning as heuristic search.

B. Bollobás.
*Random graphs.*

R. E. Bryant.
Symbolic Boolean manipulation with ordered binary decision diagrams.

Tom Bylander.
The computational complexity of propositional STRIPS planning.

Tom Bylander.
A probabilistic analysis of propositional STRIPS planning.
S. A. Cook.
The complexity of theorem proving procedures.

Joseph C. Culberson and Jonathan Schaeffer.
Searching with pattern databases.

Adnan Darwiche.
Decomposable negation normal form.

Klaus Dräger, Bernd Finkbeiner, and Andreas Podelski.
Directed model checking with distance-preserving abstractions.

Minh Binh Do and Subbarao Kambhampati.
Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP.
Adnan Darwiche and Pierre Marquis.
A knowledge compilation map.

Yannis Dimopoulos, Bernhard Nebel, and Jana Koehler.
Encoding planning problems in nonmonotonic logic programs.

G. Dueck and T. Scheuer.
Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing.
*Journal of Computational Physics, 90:161–175, 1990.*

Stefan Edelkamp.
Planning with pattern databases.
Unpublished.

Stefan Edelkamp and Frank Reffel.
OBDDs in heuristic search.


Alfonso Gerevini and Ivan Serina.
LPG: a planner based on local search for planning graphs with action costs.

Hana Galperin and Avi Wigderson.
Succinct representations of graphs.
See [Loz88] for a correction.

Malte Helmert and Carmel Domshlak.
Landmarks, critical paths and abstractions: What’s the difference anyway.

Malte Helmert, Patrik Haslum, and Joerg Hoffmann.
Flexible abstraction heuristics for optimal sequential planning.

J. Hoffmann and B. Nebel.
The FF planning system: fast plan generation through heuristic search.
A formal basis for the heuristic determination of minimum-cost paths.

Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutilier.
SPUDD: Stochastic planning using decision diagrams.

E. Hansen, R. Zhou, and Z. Feng.
Symbolic heuristic search using decision diagrams.

Paul Jackson and Daniel Sheridan.
Clause form conversions for Boolean circuits.

R. M. Jensen, M. M. Veloso, and R. E. Bryant.
State-set branching: Leveraging BDDs for heuristic search.
*Artificial Intelligence, 172(2-3):103–139, 2008.*
Optimization by simulated annealing.

R. E. Korf.
Depth-first iterative deepening: an optimal admissible tree search.

Henry Kautz and Bart Selman.
Planning as satisfiability.

Henry Kautz and Bart Selman.
Pushing the envelope: planning, propositional logic, and stochastic search.

Henry Kautz and Bart Selman.
Unifying SAT-based and graph-based planning.
Antonio Lozano and José L. Balcázar.
The complexity of graph problems for succinctly represented graphs.
In Manfred Nagl, editor, Graph-Theoretic Concepts in Computer Science, 15th
International Workshop, WG’89, number 411 in Lecture Notes in Computer Science,

Michael L. Littman.
Probabilistic propositional planning: Representations and complexity.
In Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97) and
9th Innovative Applications of Artificial Intelligence Conference (IAAI-97), pages

Antonio Lozano.
NP-hardness of succinct representations of graphs.
Bulletin of the European Association for Theoretical Computer Science, 35:158–163,

Drew McDermott.
A heuristic estimator for means-ends analysis in planning.
In Brian Drabble, editor, Proceedings of the Third International Conference on Artificial

Drew McDermott.
The Planning Domain Definition Language.
Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision
and Control, Yale University, October 1998.
References

Omid Madani, Steve Hanks, and Anne Condon.
On the undecidability of probabilistic planning and related stochastic optimization problems.

Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik.
Chaff: engineering an efficient SAT solver.

David A. McAllester and David Rosenblitt.
Systematic nonlinear planning.

David Mitchell, Bart Selman, and Hector Levesque.
Hard and easy distributions of SAT problems.

Panagiotis Manolios and Daron Vroon.
Efficient circuit to CNF conversion.
Aldo Porco, Alejandro Machado, and Blai Bonet.
Automatic polytime reductions of NP problems into a fragment of STRIPS.

S. Richter and M. Helmert.
Preferred operators and deferred evaluation in satisficing planning.

Jussi Rintanen, Keijo Heljanko, and Ilkka Niemelä.
Planning as satisfiability: parallel plans and algorithms for plan search.

Jussi Rintanen.
A planning algorithm not based on directional search.

Jussi Rintanen.
Complexity of planning with partial observability.
Jussi Rintanen.
Evaluation strategies for planning as satisfiability.

Jussi Rintanen.
Phase transitions in classical planning: an experimental study.

Jussi Rintanen.
Conditional planning in the discrete belief space.

Jussi Rintanen.
Complexity of concurrent temporal planning.
Jussi Rintanen.
Regression for classical and nondeterministic planning.

Jussi Rintanen.
Heuristic planning with SAT: beyond uninformed depth-first search.

Jussi Rintanen.
Heuristics for planning with SAT.

Matthew Streeter and Stephen F. Smith.
Using decision procedures efficiently for optimization.
References XIV

Meera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen, and Demosthenis Teneketzis.
Diagnosability of discrete-event systems.

P. H. Starke.
Reachability analysis of Petri nets using symmetries.

G. S. Tseitin.
On the complexity of derivation in propositional calculus.

Antti Valmari.
Stubborn sets for reduced state space generation.

Peter van Beek and Xinguang Chen.
CPlan: a constraint programming approach to planning.
Vincent Vidal.  
A lookahead strategy for heuristic search planning.  
In Shlomo Zilberstein, Jana Koehler, and Sven Koenig, editors, ICAPS 2004.  

Emmanuel Zarpas.  
Simple yet efficient improvements of SAT based bounded model checking.  