

Planning and Optimization

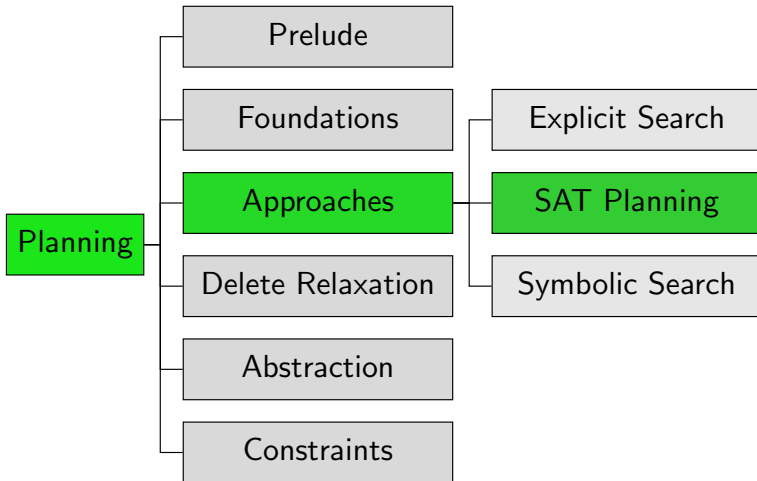
C4. SAT Planning: Core Idea and Sequential Encoding

Malte Helmert and Gabriele Röger

Universität Basel

October 14, 2024

Content of the Course



Introduction

SAT Solvers

- **SAT solvers** (algorithms that find satisfying assignments to CNF formulas) are one of the major success stories in solving hard combinatorial problems.
- Can we leverage them for classical planning?
- ↪ **SAT planning** (a.k.a. planning as satisfiability)

background on SAT Solvers:

↪ Foundations of Artificial Intelligence Course, Ch. E4–E5

Complexity Mismatch

- The SAT problem is **NP-complete**, while PLANEX is **PSPACE-complete**.
- ↪ one-shot polynomial reduction from PLANEX to SAT not possible (unless $NP = PSPACE$)

Solution: Iterative Deepening

- We can generate a propositional formula that tests if task Π has a plan with **horizon** (length bound) T in time $O(\|\Pi\|^k \cdot T)$ (\rightsquigarrow pseudo-polynomial reduction).
- Use as building block of algorithm that probes increasing horizons (a bit like IDA*).
- Can be efficient if there exist plans that are **not excessively long**.

SAT Planning: Main Loop

basic SAT Planning algorithm:

SAT Planning

```
def satplan( $\Pi$ ):  
    for  $T \in \{0, 1, 2, \dots\}$ :  
         $\varphi := \text{build\_sat\_formula}(\Pi, T)$   
         $l = \text{sat\_solver}(\varphi)$  ▷ returns a model or none  
        if  $l$  is not none:  
            return  $\text{extract\_plan}(\Pi, T, l)$ 
```

Termination criterion for unsolvable tasks?

Formula Overview

SAT Formula: CNF?

- SAT solvers require **conjunctive normal form** (CNF), i.e., formulas expressed as collection of **clauses**.
- We will make sure that our SAT formulas are in CNF when our input is a **STRIPS** task.
- We do allow fully general propositional tasks, but then the formula may need additional conversion to CNF.

SAT Formula: Variables

- given propositional planning task $\Pi = \langle V, I, O, \gamma \rangle$
- given **horizon** $T \in \mathbb{N}_0$

Variables of the SAT Formula

- propositional variables v^i for all $v \in V, 0 \leq i \leq T$
encode **state after i steps** of the plan
- propositional variables o^i for all $o \in O, 1 \leq i \leq T$
encode **operator(s) applied in i -th step** of the plan

Formulas with Time Steps

Definition (Time-Stamped Formulas)

Let φ be a propositional logic formula over the variables V .

Let $0 \leq i \leq T$.

We write φ^i for the formula obtained from φ by replacing each $v \in V$ with v^i .

Example: $((a \wedge b) \vee \neg c)^3 = (a^3 \wedge b^3) \vee \neg c^3$

SAT Formula: Motivation

We want to express a **formula** whose **models** are exactly the plans/traces with T steps.

For this, the formula must express four things:

- The variables v^0 ($v \in V$) define the initial state.
- The variables v^T ($v \in V$) define a goal state.
- We select exactly one operator variable o^i ($o \in O$) for each time step $1 \leq i \leq T$.
- If we select o^i , then variables v^{i-1} and v^i ($v \in V$) describe a state transition from the $(i-1)$ -th state of the plan to the i -th state of the plan (that uses operator o).

The final formula is the **conjunction** of all these parts.

Initial State, Goal, Operator Selection

SAT Formula: Initial State

SAT Formula: Initial State

initial state clauses:

- v^0 for all $v \in V$ with $I(v) = \mathbf{T}$
- $\neg v^0$ for all $v \in V$ with $I(v) = \mathbf{F}$

SAT Formula: Goal

SAT Formula: Goal

goal clauses:

- γ^T

For STRIPS, this is a conjunction of unit clauses.
For general goals, this may not be in clause form.

SAT Formula: Operator Selection

Let $O = \{o_1, \dots, o_n\}$.

SAT Formula: Operator Selection

operator selection clauses:

- $o_1^i \vee \dots \vee o_n^i$ for all $1 \leq i \leq T$

operator exclusion clauses:

- $\neg o_j^i \vee \neg o_k^i$ for all $1 \leq i \leq T, 1 \leq j < k \leq n$

Transitions

SAT Formula: Transitions

We now get to the interesting/challenging bit:
encoding the transitions.

Key observations: if we apply operator o at time i ,

- its **precondition** must be satisfied at time $i - 1$:

$$o^i \rightarrow pre(o)^{i-1}$$

- variable v is true at time i iff its **regression** is true at $i - 1$:

$$o^i \rightarrow (v^i \leftrightarrow regr(v, eff(o))^{i-1})$$

Question: Why $regr(v, eff(o))$, not $regr(v, o)$?

Simplifications and Abbreviations

- Let us pick the last formula apart to understand it better (and also get a CNF representation along the way).
- Let us call the formula τ (“transition”):
$$\tau = o^i \rightarrow (v^i \leftrightarrow \text{regr}(v, \text{eff}(o))^{i-1}).$$
- First, some abbreviations:
 - Let $e = \text{eff}(o)$.
 - Let $\rho = \text{regr}(v, e)$ (“regression”).
We have $\rho = \text{effcond}(v, e) \vee (v \wedge \neg \text{effcond}(\neg v, e))$.
 - Let $\alpha = \text{effcond}(v, e)$ (“added”).
 - Let $\delta = \text{effcond}(\neg v, e)$ (“deleted”).

$$\rightsquigarrow \tau = o^i \rightarrow (v^i \leftrightarrow \rho^{i-1}) \text{ with } \rho = \alpha \vee (v \wedge \neg \delta)$$

Picking it Apart (1)

Reminder: $\tau = o^i \rightarrow (v^i \leftrightarrow \rho^{i-1})$ with $\rho = \alpha \vee (v \wedge \neg\delta)$

$$\begin{aligned}\tau &= o^i \rightarrow (v^i \leftrightarrow \rho^{i-1}) \\ &\equiv o^i \rightarrow ((v^i \rightarrow \rho^{i-1}) \wedge (\rho^{i-1} \rightarrow v^i)) \\ &\equiv \underbrace{(o^i \rightarrow (v^i \rightarrow \rho^{i-1}))}_{\tau_1} \wedge \underbrace{(o^i \rightarrow (\rho^{i-1} \rightarrow v^i))}_{\tau_2}\end{aligned}$$

\rightsquigarrow consider this two **separate** constraints τ_1 and τ_2

Picking it Apart (2)

Reminder: $\tau_1 = o^i \rightarrow (v^i \rightarrow \rho^{i-1})$ with $\rho = \alpha \vee (v \wedge \neg\delta)$

$$\begin{aligned}
 \tau_1 &= o^i \rightarrow (v^i \rightarrow \rho^{i-1}) \\
 &\equiv o^i \rightarrow (\neg\rho^{i-1} \rightarrow \neg v^i) \\
 &\equiv (o^i \wedge \neg\rho^{i-1}) \rightarrow \neg v^i \\
 &\equiv (o^i \wedge \neg(\alpha^{i-1} \vee (v^{i-1} \wedge \neg\delta^{i-1}))) \rightarrow \neg v^i \\
 &\equiv (o^i \wedge (\neg\alpha^{i-1} \wedge (\neg v^{i-1} \vee \delta^{i-1}))) \rightarrow \neg v^i \\
 &\equiv \underbrace{((o^i \wedge \neg\alpha^{i-1} \wedge \neg v^{i-1}) \rightarrow \neg v^i)}_{\tau_{11}} \wedge \underbrace{((o^i \wedge \neg\alpha^{i-1} \wedge \delta^{i-1}) \rightarrow \neg v^i)}_{\tau_{12}}
 \end{aligned}$$

\rightsquigarrow consider this two **separate** constraints τ_{11} and τ_{12}

Interpreting the Constraints (1)

Can we give an **intuitive description** of τ_{11} and τ_{12} ?

Interpreting the Constraints (1)

Can we give an **intuitive description** of τ_{11} and τ_{12} ?

↪ Yes!

- $\tau_{11} = (o^i \wedge \neg \alpha^{i-1} \wedge \neg v^{i-1}) \rightarrow \neg v^i$

“When applying o , if v is false and o does not add it, it remains false.”

- called **negative frame clause**

- in clause form: $\neg o^i \vee \alpha^{i-1} \vee v^{i-1} \vee \neg v^i$

- $\tau_{12} = (o^i \wedge \neg \alpha^{i-1} \wedge \delta^{i-1}) \rightarrow \neg v^i$

“When applying o , if o deletes v and does not add it, it is false afterwards.” (Note the add-after-delete semantics.)

- called **negative effect clause**

- in clause form: $\neg o^i \vee \alpha^{i-1} \vee \neg \delta^{i-1} \vee \neg v^i$

For STRIPS tasks, these are indeed clauses. (And in general?)

Picking it Apart (3)

Almost done!

Picking it Apart (3)

Almost done!

Reminder: $\tau_2 = o^i \rightarrow (\rho^{i-1} \rightarrow v^i)$ with $\rho = \alpha \vee (v \wedge \neg\delta)$

$$\begin{aligned}\tau_2 &= o^i \rightarrow (\rho^{i-1} \rightarrow v^i) \\ &\equiv (o^i \wedge \rho^{i-1}) \rightarrow v^i \\ &\equiv (o^i \wedge (\alpha^{i-1} \vee (v^{i-1} \wedge \neg\delta^{i-1}))) \rightarrow v^i \\ &\equiv \underbrace{((o^i \wedge \alpha^{i-1}) \rightarrow v^i)}_{\tau_{21}} \wedge \underbrace{((o^i \wedge v^{i-1} \wedge \neg\delta^{i-1}) \rightarrow v^i)}_{\tau_{22}}\end{aligned}$$

\rightsquigarrow consider this two **separate** constraints τ_{21} and τ_{22}

Interpreting the Constraints (2)

How about an **intuitive description** of τ_{21} and τ_{22} ?

Interpreting the Constraints (2)

How about an **intuitive description** of τ_{21} and τ_{22} ?

- $\tau_{21} = (o^i \wedge \alpha^{i-1}) \rightarrow v^i$

“When applying o , if o adds v , it is true afterwards.”

- called **positive effect clause**
- in clause form: $\neg o^i \vee \neg \alpha^{i-1} \vee v^i$

- $\tau_{22} = (o^i \wedge v^{i-1} \wedge \neg \delta^{i-1}) \rightarrow v^i$

“When applying o , if v is true and o does not delete it, it remains true.”

- called **positive frame clause**
- in clause form: $\neg o^i \vee \neg v^{i-1} \vee \delta^{i-1} \vee v^i$

For STRIPS tasks, these are indeed clauses. (But not in general.)

SAT Formula: Transitions

SAT Formula: Transitions

precondition clauses:

- $\neg o^i \vee pre(o)^{i-1}$ for all $1 \leq i \leq T, o \in O$

positive and negative effect clauses:

- $\neg o^i \vee \neg \alpha^{i-1} \vee v^i$ for all $1 \leq i \leq T, o \in O, v \in V$
- $\neg o^i \vee \alpha^{i-1} \vee \neg \delta^{i-1} \vee \neg v^i$ for all $1 \leq i \leq T, o \in O, v \in V$

positive and negative frame clauses:

- $\neg o^i \vee \neg v^{i-1} \vee \delta^{i-1} \vee v^i$ for all $1 \leq i \leq T, o \in O, v \in V$
- $\neg o^i \vee \alpha^{i-1} \vee v^{i-1} \vee \neg v^i$ for all $1 \leq i \leq T, o \in O, v \in V$

where $\alpha = effcond(v, eff(o))$, $\delta = effcond(\neg v, eff(o))$.

For STRIPS, all except the precondition clauses are in clause form.

The precondition clauses are easily convertible to CNF

(one clause $\neg o^i \vee v^{i-1}$ for each precondition atom v of o).

Summary

Summary

- **SAT planning** (planning as satisfiability) expresses a sequence of bounded-horizon planning tasks as SAT formulas.
- Plans can be extracted from satisfying assignments; unsolvable tasks are challenging for the algorithm.
- For each **time step**, there are propositions encoding which state variables are true and which operators are applied.
- We describe a basic **sequential** encoding where one operator is applied at every time step.
- The encoding produces a **CNF** formula for **STRIPS** tasks.
- The encoding follows naturally (with some work) from using **regression** to link state variables in adjacent time steps.