Practical SAT Solving
Lecture 4
Carsten Sinz, Tomáš Balyo | May 14, 2019

INSTITUTE FOR THEORETICAL COMPUTER SCIENCE
What is Planning

Informal Definition
Planning is the process of finding a plan, i.e., a sequence of actions that changes the state of the world from some initial state to a desired (goal) state.

Examples
- Delivering some packages
- Building a submarine
- Robot motion planning
- Fulfilling a scientific goal by an autonomous space probe
Trucking Example

Initial State
- There is a truck and a package in city A
- There is a package in city B

Goal
- There are two packages in city C

Possible Actions
- (Un)loading packages from/on the truck, driving between cities
A planning problem instance $\Pi$ is a tuple $(\mathcal{X}, \mathcal{A}, s_I, s_G)$ where

- $\mathcal{X}$ is a set of multivalued variables with finite domains.
  - each variable $x \in \mathcal{X}$ has a finite possible set of values $\text{dom}(x)$
- $\mathcal{A}$ is a set actions. Each action $a \in \mathcal{A}$ is a tuple $(\text{pre}(a), \text{eff}(a))$
  - $\text{pre}(a)$ is a set of preconditions of action $a$
  - $\text{eff}(a)$ is a set of effects of action $a$
  - both are sets of equalities of the form $x = v$ where $x \in \mathcal{X}$ and $v \in \text{dom}(x)$
- $s_I$ is the initial state, it is a full assignment of the variables in $\mathcal{X}$
- $s_G$ is the set of goal conditions, it is a set of equalities (same as $\text{pre}(a)$ and $\text{eff}(a)$)
World State

A state is full assignment of the variables in \( \mathcal{X} \) (each variable \( x \in \mathcal{X} \) has exactly one value assigned from its domain \( \text{dom}(x) \)). A state can be represented as a set of equalities.

The initial state \( s_I \) is a state. A state \( s \) is a goal state if \( s_G \subseteq s \).

Applicable Actions

An action \( a \in \mathcal{A} \) is applicable in the state \( s \) if \( \text{pre}(a) \subseteq s \).

Applying an Action

When an action \( a \in \mathcal{A} \) is applied in the state \( s \) it changes to the state \( s' \) such that \( \text{eff}(a) \subseteq s' \) and the difference between \( s \) and \( s' \) is minimal (only variables used in \( \text{eff}(a) \) are changed).
A Plan

A plan for $P$ for a planning problem $\Pi = (\mathcal{X}, A, s_I, s_G)$ is sequence of actions $a_1, a_2, \ldots a_n$ such that

- $\forall i \ a_i \in A$
- let $s_1 = s_I$ and $s_{i+1} = apply(s_i, a_i)$
- $a_i$ is applicable in $s_i$
- $s_G \subseteq s_{n+1}$

If $P = \{a_1, a_2, \ldots a_n\}$ then $n$ is the length of the plan $P$.

An optimal plan is a plan of shortest length.
variables: Truck Location $T$, $dom(T) = \{A, B, C\}$, Package Locations $P_1$ and $P_2$, $dom(P_1) = dom(P_2) = \{A, B, C, T\}$

Initial state: $\{T = A, P_1 = A, P_2 = B\}$

Goal: $\{P_1 = C, P_2 = C\}$

Actions: $load(P_i, L) = (\{T = L, P_i = L\}, \{P_i = T\})$
$unload(P_i, L) = (\{T = L, P_i = T\}, \{P_i = L\})$
$drive(L_1, L_2) = (\{T = L_1\}, \{T = L_2\})$ where $i \in \{1, 2\}$ and $L, L_1, L_2 \in \{A, B, C\}$
Trucking Example

World State
- $T = A, P_1 = A, P_2 = B$
- $T = A, P_1 = T, P_2 = B$
- $T = B, P_1 = T, P_2 = B$
- $T = B, P_1 = T, P_2 = T$
- $T = C, P_1 = T, P_2 = T$
- $T = C, P_1 = C, P_2 = C$

The Plan
- $load(P_1, A)$
- $drive(A, B)$
- $load(P_2, B)$
- $drive(B, C)$
- $unload(P_1, C), unload(P_1, C)$
Sokoban Example

- **Initial State**
  - There is a worker and a bunch of boxes

- **Goal**
  - All the boxes must be in goal positions

- **Possible Actions**
  - moving with the worker
  - pushing a box

- **Forbidden**
  - to pull boxes
  - move through walls or boxes

http://wiki.pe/Sokoban
Encoding Sokoban

- Variables – For each location we have variable, the domain is WORKER, BOX, EMPTY
- Initial State – assign values based on the picture
- Goal – goal position variables have value BOX
- Actions – move and push for each possible location
- \(\text{push}(L_1, L_2, L_3) = (\{L_1 = W, L_2 = B, L_3 = E\}, \{L_1 = E, L_2 = W, L_3 = B\})\)
- \(\text{move}(L_1, L_2) = (\{L_1 = W, L_2 = E\}, \{L_1 = E, L_2 = W\})\)
Is that even possible?
Encoding Planning into CNF

- We cannot encode the existence of a plan in general
- But we can encode the existence of a plan up to some length
Encoding Planning into CNF

- We cannot encode the existence of a plan in general
- But we can encode the existence of plan up to some length

SATPLAN Algorithm

- INPUT: a planning problem $\Pi$
- OUTPUT: a plan $P$

for $m := 1, 2, \ldots$ do
  $F = \text{encodePlanExists}(\Pi, m)$
  if $\text{solver.isSat}(F)$ then
    $\text{return} \ \text{extractPlan}(\Pi, m, \text{solver.solution})$
The Task

Given a planning problem instance $\Pi = (X, A, s_i, s_G)$ and $k \in \mathbb{N}$ construct a CNF formula $F$ such that $F$ is satisfiable if and only if there is a plan of length $k$ for $\Pi$. 

We will need two kinds of variables:

- Variables to encode the actions: $a_t^i$ for each $t \in \{1, \ldots, k\}$ and $a^i \in A$.
- Variables to encode the states: $b_t^x=v$ for each $t \in \{1, \ldots, k+1\}$, $x \in X$ and $v \in \text{dom}(x)$.

In total we have $k|A| + (k+1)\sum_{x \in X} |\text{dom}(x)|$ variables.
Encoding Planning into CNF

The Task

Given a planning problem instance $\Pi = (\mathcal{X}, \mathcal{A}, s_i, s_G)$ and $k \in \mathbb{N}$ construct a CNF formula $F$ such that $F$ is satisfiable if and only if there is a plan of length $k$ for $\Pi$.

We will need two kinds of variables

- Variables to encode the actions:
  $a_i^t$ for each $t \in \{1, \ldots, k\}$ and $a_i \in \mathcal{A}$

- Variables to encode the states:
  $b_{x=v}^t$ for each $t \in \{1, \ldots, k+1\}$, $x \in \mathcal{X}$ and $v \in \text{dom}(x)$

In total we have $k|\mathcal{A}| + (k + 1) \sum_{x \in \mathcal{X}} \text{dom}(x)$ variables
We will need 8 kinds of clauses

- The first state is the initial state
- The goal conditions are satisfied in the end
- Each state variable has at least one value
- Each state variable has at most one value
- If an action is applied it must be applicable
- If an action is applied its effects are applied in the next step
- State variables cannot change without an action between steps
- At most one action is used in each step
Encoding Planning into CNF

The first state is the initial state

\[ (b^1_x = v) \]
\[ \forall (x = v) \in s_I \]  \hspace{1cm} (1)

The goal conditions are satisfied in the end

\[ (b^{n+1}_x = v) \]
\[ \forall (x = v) \in s_G \]  \hspace{1cm} (2)
Encoding Planning into CNF

Each state variable has at least one value

\[(b^t_{x=v_1} \lor b^t_{x=v_2} \lor \cdots \lor b^t_{x=v_d})\]
\[\forall x \in X, \text{dom}(x) = \{v_1, v_2, \ldots, v_d\}, \forall t \in \{1, \ldots, k + 1\}\]

Each state variable has at most one value

\[(-b^t_{x=v_i} \lor -b^t_{x=v_j})\]
\[\forall x \in X, v_i \neq v_j, \{v_i, v_j\} \subseteq \text{dom}(x), \forall t \in \{1, \ldots, k + 1\}\]
Encoding Planning into CNF

If an action is applied it must be applicable

\[ \neg a^t \lor b^t_{x=v} \]
\[ \forall a \in A, \forall (x = v) \in \text{pre}(a), \forall t \in \{1, \ldots, k\} \tag{5} \]

If an action is applied its effects are applied in the next step

\[ \neg a^t \lor b^{t+1}_{x=v} \]
\[ \forall a \in A, \forall (x = v) \in \text{eff}(a), \forall t \in \{1, \ldots, k\} \tag{6} \]
Encoding Planning into CNF

State variables cannot change without an action between steps

\[
(¬b_{x=v}^{t+1} \lor b_{x=v}^t \lor a_{s_1}^t \lor \cdots \lor a_{s_j}^t)
\]

\(\forall x \in X, \forall v \in \text{dom}(x), \text{support}(x = v) = \{a_{s_1}, \ldots, a_{s_j}\}, \forall t \in \{1, \ldots, k\}\)

(7)

By \text{support}(x = v) \subseteq \mathcal{A} we mean the set of supporting actions of the assignment \(x = v\), i.e., the set of actions that have \(x = v\) as one of their effects.
At most one action is used in each step

\[(¬a_i^t \lor ¬a_j^t)\]

\[∀\{a_i, a_j\} ⊆ A, a_i ≠ a_j \forall t ∈ \{1, \ldots, k\}\]
The Task Solved

Given a planning problem instance $\Pi = (X, A, s_I, s_G)$ and $k \in \mathbb{N}$ a CNF formula $F$, which is a conjunction of all the above described clauses is satisfiable if and only if there is plan of length $k$ for $\Pi$.

Optimizations

- Better encoding of at-most-one
- Allowing several actions in each step
- Encoding variable transitions instead of variable values
Let $M$ be a non-deterministic Turing machine that accepts an input $x$ in $P(|x|)$ time, where $P$ is a polynomial function.

- $M$ on $x$ will use at most $P(|x|)$ tape entries
- $M$ on input $x$ as a SAS+ planning problem $\Pi$
  - State variables are the state of the TM and the $P(|x|)$ tape entries
  - The transition function table is encoded as actions
  - Initial state: tape contains input, TM state is initial state
  - Goal state: TM state is an accepting state

Encode $\Pi$ for plan length $k = P(|x|)$ into a CNF formula $F_k$

- $F_k$ is SAT if and only if $M$ accepts $x$ in $P(|x|)$ time
- $F_k$ has polynomial size w.r.t. to $M$ and $x$
Planning with incremental SAT

- we are solving a sequence of similar formulas
- how do they differ?
- how to use an incremental solver in this case?
Planning with incremental SAT

- The formula $F_k$ is the subset of $F_{k+1}$ except for the goal clauses.
- The goal clauses will be added as removable (in this case, since they are unit, we can just assume them)

### Incremental SATPLAN Algorithm

<p>| INPUT: a planning problem $\Pi$ |</p>
<table>
<thead>
<tr>
<th>OUTPUT: a plan $P$</th>
</tr>
</thead>
</table>

```python
S = initSolver()
addInitialStateClauses(S)
for m := 1, 2, \ldots \ do
    addClausesForStep(m, S)
    assumeGoalConditionsAtStep(m, S)
if satisfiable(S) then return extractPlan(\Pi, m, getValues(S))
```
The DIMSPEC format

- Many other (than planning) problems have a similar structure
  - for example bounded model checking
- They can be specified using the DIMSPEC format
- DIMSPEC is four cnf formulas, where the "$p\text{ cnf }<n> <m>$" line is replaced by:
  - $i\text{ cnf }<n> <m>$ for the initial state specification ($n$ variables)
  - $g\text{ cnf }<n> <m>$ for the goal state specification ($n$ variables)
  - $u\text{ cnf }<n> <m>$ for the universal state specification ($n$ variables)
  - $t\text{ cnf }<n> <m>$ for the specification of the transition (between two neighboring states) ($2n$ variables)
The DIMSPEC format example

c this is an example of a dimspec file
i cnf 5 3
-1 2 0
2 3 -5 0
4 0
g cnf 5 1
5 0
u cnf 5 2
-1 2 3 0
-3 4 5 0
t cnf 10 2
-2 7 8 0
-4 9 10 0
Planning as DIMSPEC

- Initial state specification clauses: \((b_{x=v})\) added \(\forall (x = v) \in S_I\)
- Goal state specification clauses: \((b_{x=v})\) added \(\forall (x = v) \in S_G\)
- Universal state specification clauses:
  - \((b_{x=v_1} \lor b_{x=v_2} \lor \cdots \lor b_{x=v_d})\) added \(\forall x \in X\) where \(\text{dom}(x) = \{v_1, v_2, \ldots, v_d\}\) – at least one value
  - \((b_{x=i} \lor b_{x=j})\) added \(\forall x \in X\) \(i \neq j \in \text{dom}(x)\) – at most one value
  - \((\overline{a} \lor b_{x=v})\) added \(\forall a \in A, \forall (x = v) \in \text{pre}(a)\) – action preconditions
  - \((\overline{a_i} \lor \overline{a_j})\) added \(\forall i \neq j\) – at most one action
- Transition specification clauses
  - \((\overline{a} \lor b'_{x=v})\) added \(\forall a \in A, \forall (x = v) \in \text{eff}(a)\) – action effects
  - \((b'_{x=v} \lor b_{x=v} \lor a_{s_1} \lor \cdots \lor a_{s_j})\) added \(\forall x \in X, \forall v \in \text{dom}(x)\) where \(\text{support}(x = v) = \{a_{s_1}, \ldots, a_{s_j}\}\) – values cannot change without a reason
Solving DIMSPEC

- Same as solving planning with incremental SAT

The Basic DISMPEC Solving Algorithm

- INPUT: a DIMSPEC problem
- OUTPUT: a truth assignment

\[ S = \text{initSolver()} \]
\[ \text{addInitialStateClauses}(S) \]
\[ \text{for } m := 1, 2, \ldots \text{ do} \]
\[ \quad \text{addUniversalConditionsWithRenaming}(m, S) \]
\[ \quad \text{if } m > 1 \text{ then } \text{addTransitionalConditionsWithRenaming}(m, S) \]
\[ \quad \text{assumeGoalConditionsWithRenaming}(m, S) \]
\[ \quad \text{if satisfiable}(S) \text{ then } \text{return} \text{ getValues}(S) \]