Chapter 10

Control Rules in Planning
Motivation

- Often, planning can be done much more efficiently if we have domain-specific information.
- Example:
  - Classical planning is EXPSPACE-complete.
  - Block-stacking can be done in time $O(n^3)$.
- But we don’t want to have to write a new domain-specific planning system for each problem!

- *Domain-configurable* planning algorithm:
  - Domain-independent search engine.
  - Input includes the pruning rules for the domain.
Outline

- Language for writing domain-specific pruning rules
  - Based on modal logic

- Domain-configurable planning algorithm
  - Input includes the pruning rules for the domain

- Example: block stacking

```
Abstract-search(u)
  if Terminal(u) then return(u)
  u ← Refine(u) ;; refinement step
  B ← Branch(u) ;; branching step
  B' ← Prune(B) ;; pruning step
  if B' = ∅ then return(failure)
  nondeterministically choose v ∈ B'
  return(Abstract-search(v))
end
```

- Presentation similar to the chapter, but not identical
  - Based partly on TLPlan [Bacchus & Kabanza 2000]
First-order Logic and Modal Logics

- First Order Logic (FOL):
  - constant symbols, function symbols, predicate symbols
  - logical connectives (\( \lor, \land, \neg, \Rightarrow, \Leftrightarrow \)) and quantifiers (\( \forall, \exists \))
  - e.g., \( \text{on}(A,B) \land \text{on}(B,C) \)
    \( \exists x \ \text{on}(x,A) \)
    \( \forall x (\text{ontable}(x) \Rightarrow \text{clear}(x)) \)

- Model:
  - For our purposes, a world-state \( s \) that the formulas refer to
    - This is what gives the formulas meaning
  - \( s \models \text{on}(A,B) \) read “\( s \) satisfies \( \text{on}(A,B) \)” or “\( s \) models \( \text{on}(A,B) \)”
    - means that \( \text{on}(A,B) \) is true in the state \( s \)

- Modal logic: FOL plus \textit{modal operators}
  - to express concepts that would be difficult to express within FOL
Linear Temporal Logic

- **Linear Temporal Logic (LTL):**
  - Time is a sequence of instants 1, 2, 3, …
  - There is a sequence of states $\mathcal{M}=\langle s_0, s_1, \ldots \rangle$

- **Modal operators to refer to the states in which formulas are true:**
  - $\bigcirc f$ - *next* $f$ - $f$ holds in the next state, e.g., $\bigcirc \text{on}(A,B)$
  - $\lozenge f$ - *eventually* $f$ - $f$ either holds now or in some future state
  - $\Box f$ - *always* $f$ - $f$ holds now and in all future states
  - $f_1 \mathbf{U} f_2$ - $f_1$ until $f_2$ - $f_2$ either holds now or in some future state, and $f_1$ holds until then

- **Propositional constant symbols** TRUE and FALSE
Linear Temporal Logic (continued)

- Quantifiers cause problems with computability
  - Suppose \( f(x) \) is true for infinitely many values of \( x \)
  - Problem evaluating truth of \( \forall x \ f(x) \) and \( \exists x \ f(x) \)

- Bounded quantifiers
  - Let \( g(x) \) be such that \( \{ x : g(x) \} \) is finite and easily computed
    \[
    \forall [x:g(x)] \ f(x)
    \]
    - means \( \forall x \ (g(x) \Rightarrow f) \)
    - expands into \( f(x_1) \land f(x_2) \land \ldots \land f(x_n) \)
  
  \[
  \exists [x:g(x)] \ f(x)
  \]
    - means \( \exists x \ (g(x) \land f) \)
    - expands into \( f(x_1) \lor f(x_2) \lor \ldots \lor f(x_n) \)
Models for LTL

- A model is a triple \((M, s_i, V)\)
  - \(M = \langle s_0, s_1, \ldots \rangle\) is a sequence of states
  - \(s_i\) is the \(i\)'th state in \(M\),
  - \(V\) is a variable assignment function
    - a substitution that maps all variables into constants

- Write \((M, s_i, V) \models f\) to mean that \(V(f)\) is true in \(s_i\)

- Always require that
  \((M, s_i, V) \models \text{TRUE}\)
  \((M, s_i, V) \models \neg \text{FALSE}\)
Examples

- \( (M,s_0,V) \models \Box \Box \text{on}(A,B) \)
  - \( A \) is on \( B \) in \( s_2 \), the 2nd state after \( s_0 \)

- Abbreviations:
  - \( (M,s_0) \models \Box \Box \text{on}(A,B) \) no free variables, so \( V \) is irrelevant
  - \( M \models \Box \Box \text{on}(A,B) \) if no state specified, \( s_0 \) is the default

- \( M \models \Box \neg \text{holding}(C) \)
  - in every state in \( M \), we aren’t holding \( C \)

- \( M \models \Box (\text{on}(B, C) \Rightarrow (\text{on}(B, C) \cup \text{on}(A, B))) \)
  - whenever we enter a state in which \( B \) is on \( C \), \( B \) remains on \( C \) until \( A \) is on \( B \).
Augment the models to include a set of goal states $G$

$\text{GOAL}(f)$ - says $f$ is true in every $s$ in $G$.

$((M,s,I),G) \models \text{GOAL}(f) \iff (M,s,I) \models f$ for every $s \in G$
Blocks World - Example

- Blocks-world operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Preconditions and Deletes</th>
<th>Adds</th>
</tr>
</thead>
<tbody>
<tr>
<td>pickup(x)</td>
<td>onto(x), clear(x), handempty.</td>
<td>holding(x).</td>
</tr>
<tr>
<td>putdown(x)</td>
<td>holding(x).</td>
<td>ontable(x), clear(x), handempty.</td>
</tr>
<tr>
<td>stack(x, y)</td>
<td>holding(x), clear(y).</td>
<td>on(x, y), clear(x), handempty.</td>
</tr>
<tr>
<td>unstack(x, y)</td>
<td>on(x, y), clear(x), handempty.</td>
<td>holding(x), clear(y).</td>
</tr>
</tbody>
</table>

A planning problem:

Initial State

```
E
A
B
C
D
F
```

Goal State

```
D
A
B
C
```
Blocks World - Example

- Basic idea:
  - A *goodtower* is one in which no blocks will ever need to be moved
  - Axioms to support this:

\[
goodtower(x) \iff clear(x) \land \neg \text{GOAL}(holding(x)) \land goodtowerbelow(x)
\]

\[
goodtowerbelow(x) \iff ((ontable(x) \land \neg \exists[y:\text{GOAL}(on(x, y))]) \lor
\exists[y:on(x, y)] \neg \text{GOAL}(ontable(x)) \land \neg \text{GOAL}(holding(y)) \land \neg \text{GOAL}(clear(y))
\land \forall[z:\text{GOAL}(on(x, z))] z = y \land \forall[z:\text{GOAL}(on(z, y))] z = x
\land goodtowerbelow(y))]
\]

- Unstacking B from C would violate this axiom
- B and C are good towers

```
Initial State

```
```
Goal State
D
A
B
C
E
A
B
C
D
F
```
Blocks World Example (continued)

- Three different control rules:

  - (1) Every good tower must always remain a good tower

  \[ \square \left( \forall [x: \text{clear}(x)] \Rightarrow \bigcirc (\text{clear}(x) \lor \exists [y: \text{on}(y, x)] \text{goodtower}(y)) \right) \]

  - Unstacking B from C or putting anything but A on B will violate this.
  - Also, moving F, which is an irrelevant good tower will violate (1).
But, what about bad towers?

\[ \text{badtower}(x) \Leftrightarrow [\text{clear}(x) \land \neg \text{goodtower}(x)] \]

(2) Never put anything onto a badtower

\[ \Box \left( \forall [x: \text{clear}(x)] \ \text{goodtower}(x) \Rightarrow \circ (\text{clear}(x) \lor \exists [y: \text{on}(y, x)] \ \text{goodtower}(y) \land \text{badtower}(x) \Rightarrow \circ (\neg \exists [y: \text{on}(y, x)]) \right) \]

- with this a good sequence can only pickup blocks on top of bad towers
- the tower of blocks under E is a bad tower; so any action that stacks a block on E will violates the second conjunction of (2).
Blocks World Example (continued)

- (2) does not rule out all useless actions.
- In general, there is no point in picking up singleton bad tower blocks unless their final position is ready.
- There is no point in picking up D until we have stacked A on B.


- (3) never pick up a block from the table unless you can put it onto a good tower

\[ \square (\forall [x: clear(x)] \text{goodtower}(x) \Rightarrow \circ (clear(x) \lor \exists [y: on(y, x)] \text{goodtower}(y)) \]
\[ \land \text{badtower}(x) \Rightarrow \circ (\neg \exists [y: on(y, x)]) \]
\[ \land (\text{ontable}(x) \land \exists [y: \text{GOAL(on(x, y))}] \neg \text{goodtower}(y)) \]
\[ \Rightarrow \circ (\neg \text{holding}(x)) \]
The TLPlan procedure:

- Forward state-space search
- At each state, check whether the current path can lead to a plan in which every state satisfies the state’s control formula

Each state’s control formula is determined by *progression*

- Let $s$ be a state, $f$ be its control formula, and $s^+$ be any child of $s$
- Then the control formula for $s^+$ is $f^+ = \text{Progress}(s, f)$
  
  » Progression is defined on the next page
Procedure $\text{Progress}(f, s)$

Case

1. $f = \phi \in \mathcal{L}$ (i.e., $\phi$ contains no temporal modalities):
   
   $f^+ := \text{TRUE}$ if $s \models f$, \text{FALSE} otherwise.

2. $f = f_1 \land f_2$:
   
   $f^+ := \text{Progress}(f_1, s) \land \text{Progress}(f_2, s)$

3. $f = \neg f_1$:
   
   $f^+ := \neg \text{Progress}(f_1, s)$

4. $f = \Diamond f_1$:
   
   $f^+ := f_1$

5. $f = f_1 \lor f_2$:
   
   $f^+ := \text{Progress}(f_2, s) \lor (\text{Progress}(f_1, s) \land f)$

6. $f = \Diamond f_1$:
   
   $f^+ := \text{Progress}(f_1, s) \lor f$

7. $f = \Box f_1$:
   
   $f^+ := \text{Progress}(f_1, s) \land f$

8. $f = \forall [x : g(x)] f_1$:
   
   $f^+ := \land \left\{ \text{Progress}(\theta(f_1), s) : s \models g(c) \right\}$

9. $f = \exists [x : g(x)] f_1$:
   
   $f^+ := \lor \left\{ \text{Progress}(\theta(f_1), s) : s \models g(c) \right\}$

   where $\theta = \{x \leftarrow c\}$

Boolean simplification rules:

1. $[\text{FALSE} \land \phi] \land \phi \land \text{FALSE} \leftrightarrow \text{FALSE}$,

2. $[\text{TRUE} \land \phi] \land \phi \land \text{TRUE} \leftrightarrow \phi$,

3. $\neg \text{TRUE} \leftrightarrow \text{FALSE}$,

4. $\neg \text{FALSE} \leftrightarrow \text{TRUE}$. 
Examples

- Suppose $f = \square \text{on}(A,B)$
  - $f^+ = \text{Progress}(\text{on}(A,B), s) \land \square \text{on}(A,B)$
  - If $\text{on}(A,B)$ is true in $s$ then
    - $f^+ = \text{TRUE} \land \square \text{on}(A,B)$
    - simplifies to $\square \text{on}(A,B)$
  - If $\text{on}(A,B)$ is false in $s$ then
    - $f^+ = \text{FALSE} \land \square \text{on}(A,B)$
    - simplifies to $\text{FALSE}$

- $\square$ generates a test on the current state and propagates the test to the next state
Examples (continued)

- Suppose $f = \Box (\text{on}(A,B) \Rightarrow \Diamond \text{clear}(A))$
  - $f^+ = \text{Progress}[\Box (\text{on}(A,B) \Rightarrow \Diamond \text{clear}(A)), s]$
  - $= \text{Progress}[\text{on}(A,B) \Rightarrow \Diamond \text{clear}(A), s] \land \Box (\text{on}(A,B) \Rightarrow \Diamond \text{clear}(A))$
  - If $\text{on}(A,B)$ is true in $s$, then
    - $f^+ = \text{clear}(A) \land \Box (\text{on}(A,B) \Rightarrow \Diamond \text{clear}(A))$
      - Since $\text{on}(A,B)$ is true in the current state, the next state must satisfy $\text{clear}(A)$
      - The “always” constraint is propagated to the next state
    - $f^+$ simplifies to $\text{clear}(A) \land \Box (\text{on}(A,B) \Rightarrow \Diamond \text{clear}(A))$
  - If $\text{on}(A,B)$ is false in $s$, then
    - $f^+ = \Box (\text{on}(A,B) \Rightarrow \Diamond \text{clear}(A))$
    - The constraint is simply propagated to the next state
Procedure TLPLAN( s, f, G, A, P)
    if s satisfies G then return P
    Let $f^+ = \text{Progress}(f, s)$
    if $f^+$ is FALSE return failure
    choose an action $a$ from the set of actions $A$ whose preconditions are satisfied in $s$
    if no such action exists return failure.
    Let $s^+$ be the world that arises from applying $a$ to $s$
    return TLPLAN( $s^+$, $f^+$, G, A, P.a)

- Nondeterministic forward search
- Input includes a control formula $f$ for the problem domain
- When we expand a state $s$, we progress its formula $f$ through $s$, generating a new formula $f^+$
- If $f^+$ is false, $s$ is a dead-end
- Otherwise, $f^+$ is the control formula for all of $s$’s children
Example

Never pick up a block $x$ if $x$ is not required to be on another block $y$

\[ \square \left( \forall [x:\text{clear}(x)] \text{ontable}(x) \land \neg \exists [y:\text{GOAL}(\text{on}(x, y))] \Rightarrow \bigcirc (\neg \text{holding}(x)) \right) \]

Suppose that

\[ s = \{ \text{ontable}(a), \text{ontable}(b), \text{clear}(a), \text{clear}(b) \} \]
\[ g = \{ \text{on}(b, a) \} \]

For each clear block $x$ in $s$, evaluate

\[ \text{ontable}(x) \land \neg \exists [y:\text{GOAL}(\text{on}(x, y))] \Rightarrow \bigcirc (\neg \text{holding}(x)) \]

Progressed formula:

\[ \neg \text{holding}(a) \land \square \left( \forall [x:\text{clear}(x)] \text{ontable}(x) \land \neg \exists [y:\text{GOAL}(\text{on}(x, y))] \Rightarrow \bigcirc (\neg \text{holding}(x)) \right) \]
Control 1 fails on 1 problem of size 11

Blocks-World Results

No Control (breadth-first)
Control 1
Control 2
Control 3
Control 3 (breadth-first)
Blocks-World Results

SatPlan fails on 3 problems of size 10

UCPOP fails on all problems of size 6

BlackBox fails on 1 problem of size 10

IPP fails on 2 problems of size 11 and 12 exceeds 1GB RAM on problems of size 13
Logistics- Domain Results

- IPP fails on problems of size > 9
- BlackBox fails on problems of size > 15
- Satplan fails on 2 problems of size 14 and 15
Discussion

- 2000 International Planning Competition
  - TALplanner: same kind of algorithm, different temporal logic
    » received the top award for a “hand-tailored” planner
      • *domain-configurable* is probably a better term
  - TLPlan won the same award in the 2002 International Planning Competition
- Both of them:
  - Ran several orders of magnitude faster than the “fully automated” planners
    » especially on large problems
  - Solved problems on which the fully-automated planners ran out of time/memory