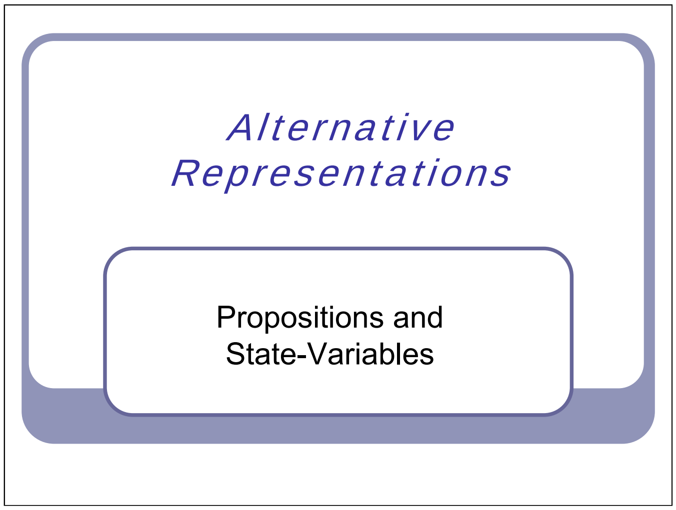


Alternative Representations

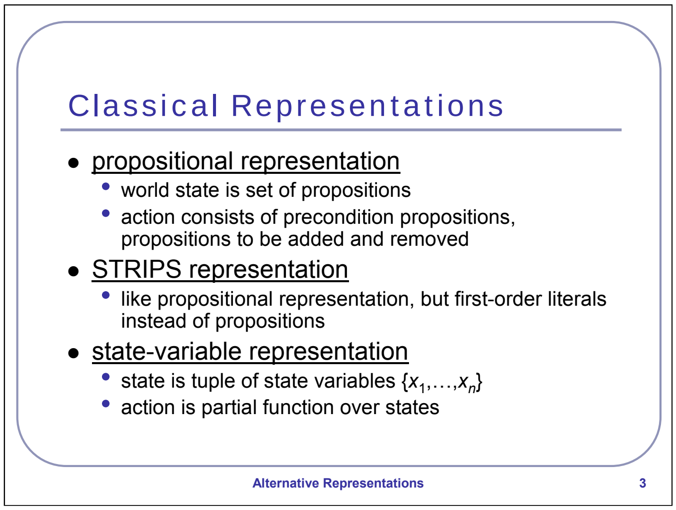
•Propositions and State-Variables

Literature

- Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning – Theory and Practice*, chapter 2. Elsevier/Morgan Kaufmann, 2004.

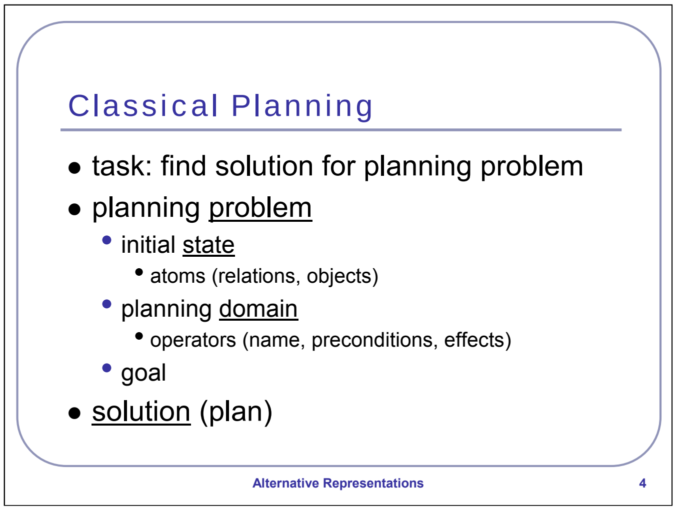
Literature

•Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning – Theory and Practice*, chapter 2. Elsevier/Morgan Kaufmann, 2004.



Classical Representations

- **<u>propositional representation</u>**
  - **world state is set of propositions**
  - **action consists of precondition propositions, propositions to be added and removed**
- **<u>STRIPS representation</u>**
  - named after STRIPS planner
  - **like propositional representation, but first-order literals instead of propositions**
  - most popular for restricted state-transitions systems
- **<u>state-variable representation</u>**
  - **state is tuple of state variables $\{x_1,\ldots,x_n\}$**
  - **action is partial function over states**
  - useful where state is characterized by attributes over finite domains
- equally expressive: planning domain in one representation can also be represented in the others

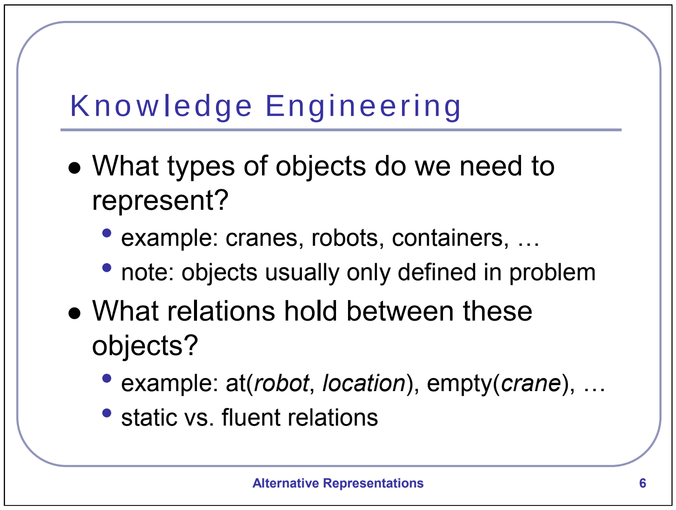


Classical Planning

- **task: find solution for planning problem**
- **planning problem**
  - **initial state**
    - state is a set of **atoms (relations, objects)**
    - difference between representations: what constitutes an atom
  - **planning domain**
    - **operators (name, preconditions, effects)**
  - **goal**
- **solution (plan)**

Overview

- **World States**
- **Domains and Operators**
- **Planning Problems**
- **Plans and Solutions**
- **Expressiveness**



Knowledge Engineering

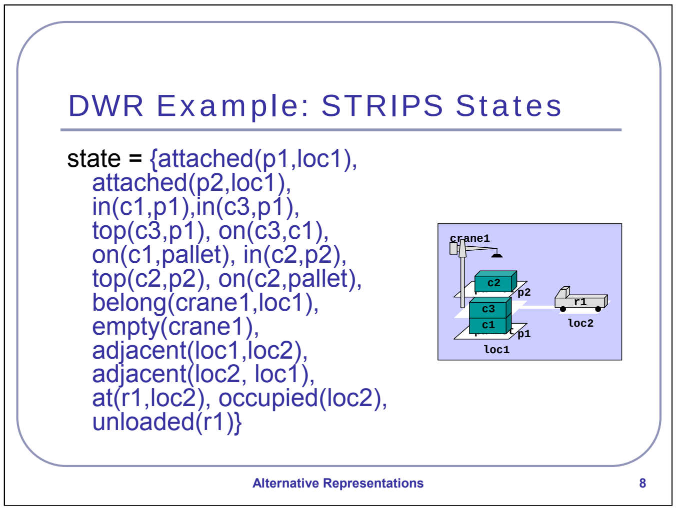
- **What types of objects do we need to represent?**
    - **example: cranes, robots, containers, …**
    - **note: objects usually only defined in problem**
        - type hierarchy usually not found in planning domain (in PDDL) but ontology is very important for KE
- **What relations hold between these objects?**
    - **example: at(*robot*, *location*), empty(*crane*), …**
        - define skeleton for readability (optional in PDDL)
    - **static vs. fluent relations**

Representing World States

	STRIPS	propositional	state-variable
state	set of atoms		
atom	first-order atom	proposition	state-variable expression
relations	yes	no	functions
objects/types	yes/maybe	no/no	yes/maybe
static relations	yes	not necessary	no

Representing World States

•states are sets of atoms in all cases; difference lies in what is an atom

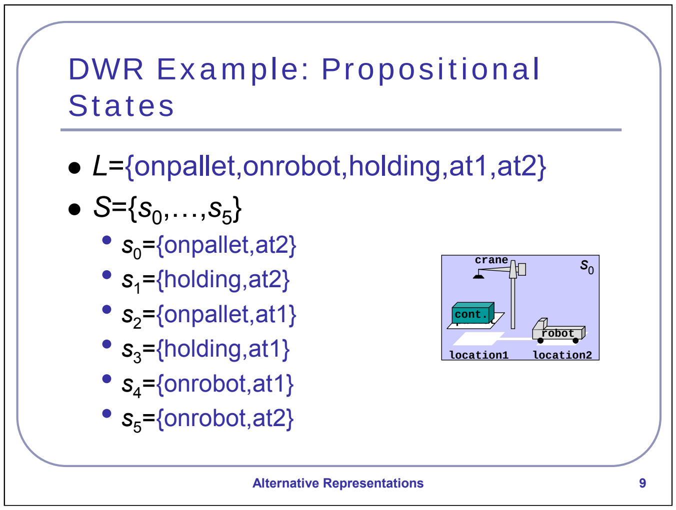


DWR Example: STRIPS States

•predicate symbols: relations for DWR domain

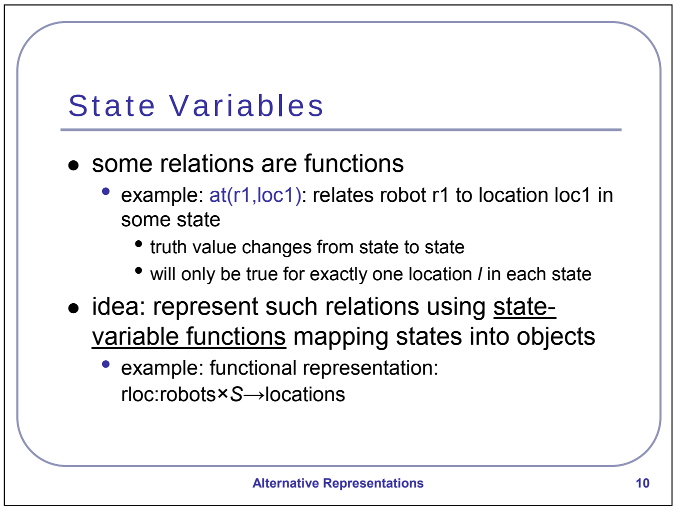
•constant symbols: for objects in the domain {loc1, loc2, r1, crane1, p1, p2, c1, c2, c3, pallet}

•state = {attached(p1,loc1), attached(p2,loc1), in(c1,p1),in(c3,p1), top(c3,p1), on(c3,c1), on(c1,pallet), in(c2,p2), top(c2,p2), on(c2,pallet), belong(crane1,loc1), empty(crane1), adjacent(loc1,loc2), adjacent(loc2, loc1), at(r1,loc2), occupied(loc2), unloaded(r1)}



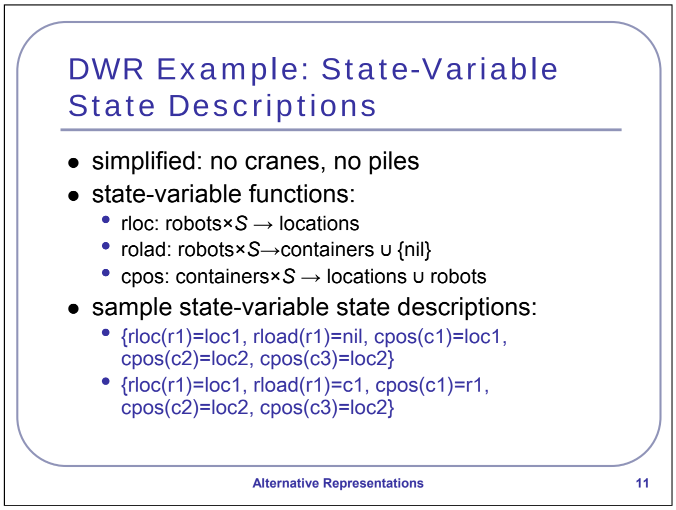
DWR Example: Propositional States

- **L={onpallet,onrobot,holding,at1,at2}**
  - meaning: container is on the ground, container on the robot, crane is holding the container, robot is at location1, robot is at location2
- **S={s_0,…,s_5}**
  - as shown in graph
  - **s_0={onpallet,at1}**
  - **s_1={holding,at1}**
  - **s_2={onpallet,at1}**
  - **s_3={holding,at1}**
  - **s_4={onrobot,at1}**
  - **s_5={onrobot,at2}**



State Variables

- **some relations are functions**
  - **example: at(r1,loc1): relates robot r1 to location loc1 in some state**
    - **truth value changes from state to state**
    - **will only be true for exactly one location *l* in each state**
      - STRIPS state containing at(r1,loc1) and at(r1,loc2) usually inconsistent
- **idea: represent such relations using <u>state-variable functions</u> mapping states into objects**
  - advantage: reduces possibilities for inconsistent states, smaller state space
  - **example: functional representation: rloc:robots×*S*→locations**
    - in general: maps objects and state into object
    - rloc is state-variable symbol that denotes state-variable function



DWR Example: State-Variable State Descriptions

- **simplified: no cranes, no piles**
  - robots can load and unload containers autonomously
- **state-variable functions:**
  - **rloc: robots$\times S \rightarrow$ locations**
    - location of a robot in a state
  - **rolad: robots$\times S \rightarrow$containers $\cup$ {nil}**
    - what a robot has loaded in a state; nil for nothing loaded
  - **cpos: containers$\times S \rightarrow$ locations $\cup$ robots**
    - where a container is in a state; at a location or on some robot
- sample state-variable state descriptions:
  - {rloc(r1)=loc1, rload(r1)=nil, cpos(c1)=loc1, cpos(c2)=loc2, cpos(c3)=loc2}
  - {rloc(r1)=loc1, rload(r1)=c1, cpos(c1)=r1, cpos(c2)=loc2, cpos(c3)=loc2}

Overview

- World States
- ➧ Domains and Operators
- Planning Problems
- Plans and Solutions
- Expressiveness

Overview

➧**World States**

•**Domains and Operators**

•**Planning Problems**

•**Plans and Solutions**

•**Expressiveness**

Knowledge Engineering

- What types of actions are there?
  - example: move robots, load containers, …
- For each action type, and each relation, what must (not) hold for the action to be applicable?
  - preconditions
- For each action type, and each relation, what relations will (no longer) hold due to the action?
  - effects (must be consistent)
- For each action type, what objects are involved in performing the action?
  - any object mentioned in the preconditions and effects
  - preconditions should mention all objects

Knowledge Engineering

•What types of actions are there?

•example: move robots, load containers, …

•For each action type, and each relation, what must (not) hold for the action to be applicable?

•preconditions

•For each action type, and each relation, what relations will (no longer) hold due to the action?

•effects (must be consistent)

•For each action type, what objects are involved in performing the action?

•any object mentioned in the preconditions and effects

•preconditions should mention all objects

Representing Operators

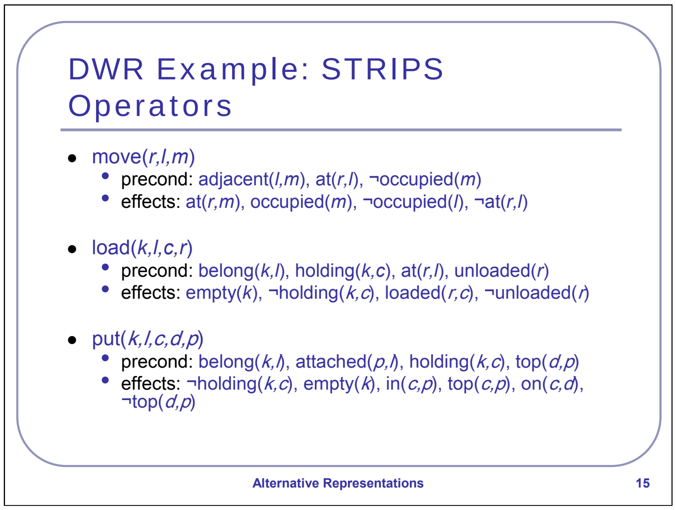
	STRIPS	propositional	state-variable
name	$n(x_1,\ldots,x_k)$	*name*	$n(x_1,\ldots,x_k)$
preconditions (set of)	first-order literals	propositions	state-variable expressions
applicability	$\text{precond}^+(a) \subseteq s \wedge \text{precond}^-(a) \cap s = \{\}$	$\text{precond}(a) \subseteq s$	$\text{precond}(a) \subseteq s$
effects (set of)	first-order literals	propositional literals	$x_s \leftarrow v$
$\gamma(s,a)$	$(s - \text{effects}^-(a)) \cup \text{effects}^+(a)$	$(s - \text{effects}^-(a)) \cup \text{effects}^+(a)$	$\{x_s=c \mid x \in X\}$ where $x_s \leftarrow c \in \text{effects}(a)$ or $x_s=c \in s$ otherwise

Alternative Representations 14

Representing Operators

•preconditions and effects essentially sets of atoms again (where atoms are different per representation)

- •propositional representation allows only for positive preconditions
- •state-variable representation only allows for equality in preconditions, no inequality
- •effects: positive and negative in all cases



DWR Example: STRIPS Operators

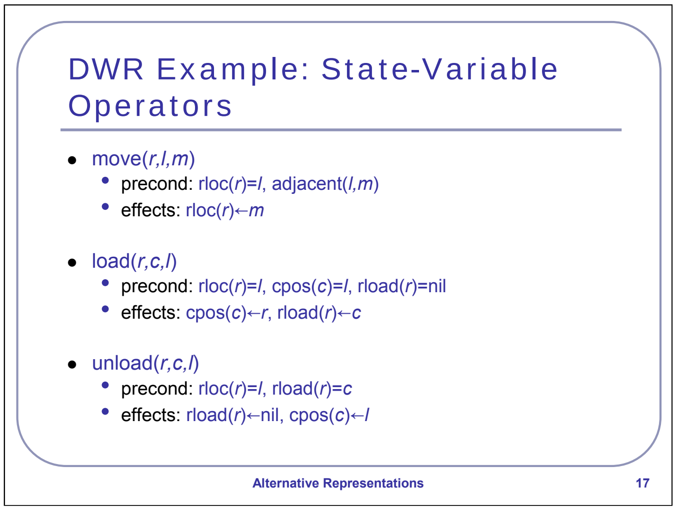
- **move(*r*,*l*,*m*)**
  - robot *r* moves from location *l* to an adjacent location *m*
  - **precond: adjacent(*l*,*m*), at(*r*,*l*), ¬occupied(*m*)**
  - **effects: at(*r*,*m*), occupied(*m*), ¬occupied(*l*), ¬at(*r*,*l*)**
- **load(*k*,*l*,*c*,*r*)**
  - crane *k* at location *l* loads container *c* onto robot *r*
  - **precond: belong(*k*,*l*), holding(*k*,*c*), at(*r*,*l*), unloaded(*r*)**
  - **effects: empty(*k*),** ¬holding(*k*,*c*), loaded(*r*,*c*), ¬unloaded(*r*)
- put(*k*,*l*,*c*,*d*,*p*)
  - crane *k* at location *l* puts container *c* onto *d* in pile *p*
  - precond: belong(*k*,*l*), attached(*p*,*l*), holding(*k*,*c*), top(*d*,*p*)
  - effects: ¬holding(*k*,*c*), empty(*k*), in(*c*,*p*), top(*c*,*p*), on(*c*,*d*), ¬top(*d*,*p*)
- similar: unload and take operators
- action: just substitute variables with values consistently

DWR Example: Propositional Actions

a	precond(a)	effects$^-$(a)	effects$^+$(a)
take	{onpallet}	{onpallet}	{holding}
put	{holding}	{holding}	{onpallet}
load	{holding,at1}	{holding}	{onrobot}
unload	{onrobot,at1}	{onrobot}	{holding}
move1	{at2}	{at2}	{at1}
move2	{at1}	{at1}	{at2}

DWR Example: Propositional Actions

- ***a*** : precond(a), effects$^-$(a), effects$^+$(a)
    - a is action name
- **take : {onpallet}, {onpallet}, {holding}**
- **put : {holding}, {holding}, {onpallet}**
- **load : {holding,at1}, {holding}, {onrobot}**
- **unload : {onrobot,at1}, {onrobot}, {holding}**
- **move1 : {at2}, {at2}, {at1}**
- **move2 : {at1}, {at1}, {at2}**



DWR Example: Operators

- simplified domain: no piles, no cranes – only three operators:
- **move(r,l,m)**
  - move robot r from location l to adjacent location m
  - **precond: rloc(r)=l, adjacent(l,m)**
    - adjacent: rigid relation
  - **effects: rloc(r)←m**
- **load(r,c,l)**
  - robot r loads container c at location l
  - **precond: rloc(r)=l, cpos(c)=l, rload(r)=nil**
  - **effects: cpos(c)←r, rload(r)←c**
- unload(r,c,l)
  - robot r unloads container c at location l
  - precond: **rloc(r)=l, rload(r)=c**
  - effects: **rload(r)←nil, cpos(c)←l**

Overview

World States

•Domains and Operators

•Planning Problems

•Plans and Solutions

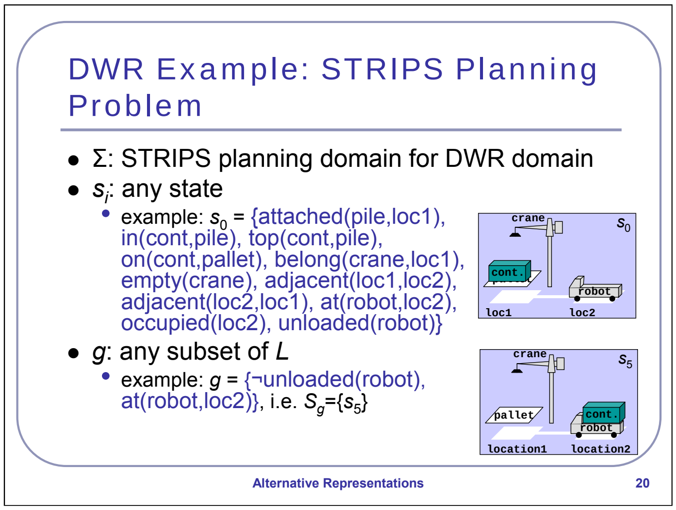
•Expressiveness

Representing Planning Problems

	STRIPS	propositional	state-variable
initial state	world state in respective representation		
domain	domain (set of operators) in respective representation		
goal	same as preconditions in respective representation		

Representing Planning Problems

•essentially the same for all representations



DWR Example: STRIPS Planning Problem

•Σ: STRIPS planning domain for DWR domain

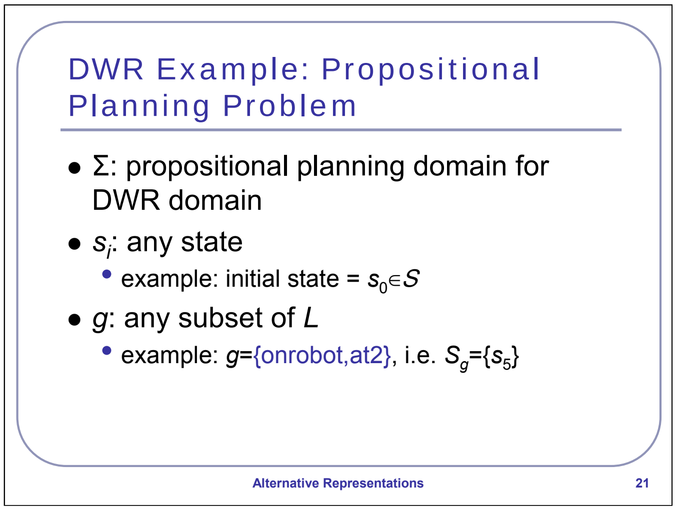
- •see previous slides

•s_i: any state

- **•example: s_0 = {attached(pile,loc1), in(cont,pile), top(cont,pile), on(cont,pallet), belong(crane,loc1), empty(crane), adjacent(loc1,loc2), adjacent(loc2,loc1), at(robot,loc2), occupied(loc2), unloaded(robot)}**
- •note: s_0 is not necessarily initial state

•g: any subset of L

- **•example: g = {¬unloaded(robot), at(robot,loc2)}, i.e. S_g={s_5}**
  - •other relations will hold, but they are not mentioned in the goal = partial specification of a state



DWR Example: Propositional Planning Problem

- **Σ: propositional planning domain for DWR domain**
  - see previous slides
- **s_i: any state**
  - **example: initial state = $s_0 \in S$**
  - note: s_0 is not necessarily initial state
- **g: any subset of L**
  - **example: g={onrobot,at2}, i.e. $S_g=\{s_5\}$**

Overview

- World States
- Domains and Operators
- Planning Problems
- ➧ Plans and Solutions
- Expressiveness

Overview

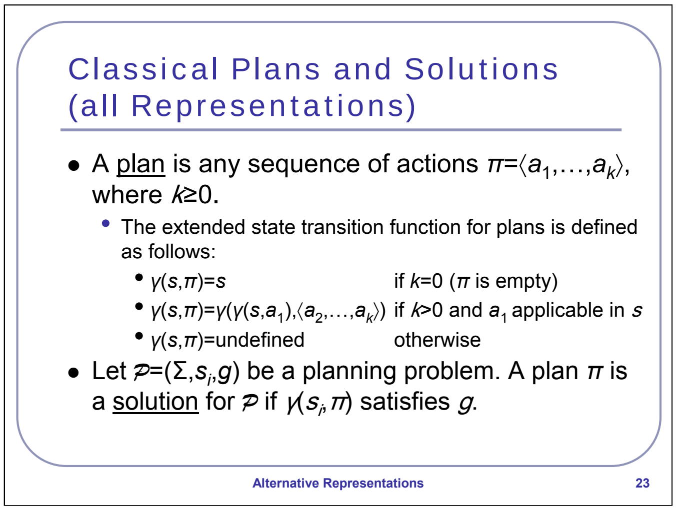
➧**World States**

•**Domains and Operators**

•**Planning Problems**

•**Plans and Solutions**

•**Expressiveness**



Classical Plans

•note: classical definitions apply to all representations

•**A <u>plan</u> is any sequence of actions $\pi=\langle a_1,\ldots,a_k\rangle$**, where $k\geq 0$.

- •k=0 means no actions in the empty plan
- •The <u>length of plan π</u> is **$|\pi|=k$, the number of actions.**
- •**If $\pi_1=\langle a_1,\ldots,a_k\rangle$** and **$\pi_2=\langle a'_1,\ldots,a'_j\rangle$** are plans, then their <u>concatenation</u> is the plan **$\pi_1\bullet\pi_2=\langle a_1,\ldots,a_k,a'_1,\ldots,a'_j\rangle$**.
- •The extended state transition function for plans is defined as follows:
  - •**$\gamma(s,\pi)$=s if k=0 (π is empty)**
  - •**$\gamma(s,\pi)=\gamma(\gamma(s,a_1),\langle a_2,\ldots,a_k\rangle)$** if k>0 and **a_1** applicable in ***s***
  - •**$\gamma(s,\pi)$=undefined otherwise**

•plan corresponds to a path through the state space

Overview

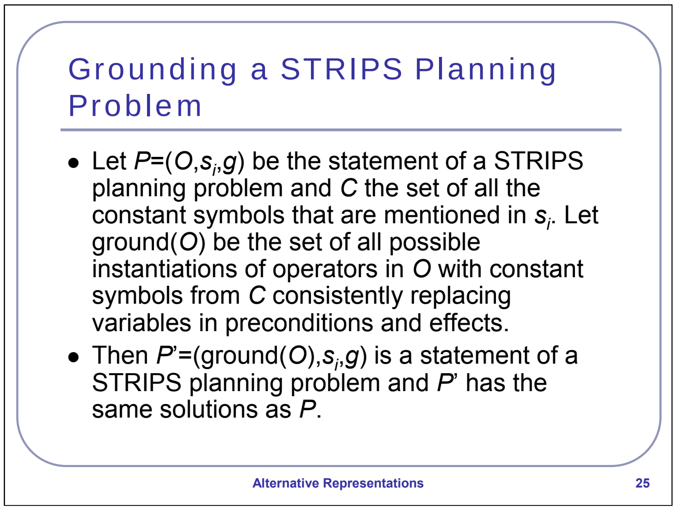
➡World States

•Domains and Operators

•Planning Problems

•Plans and Solutions

•Expressiveness



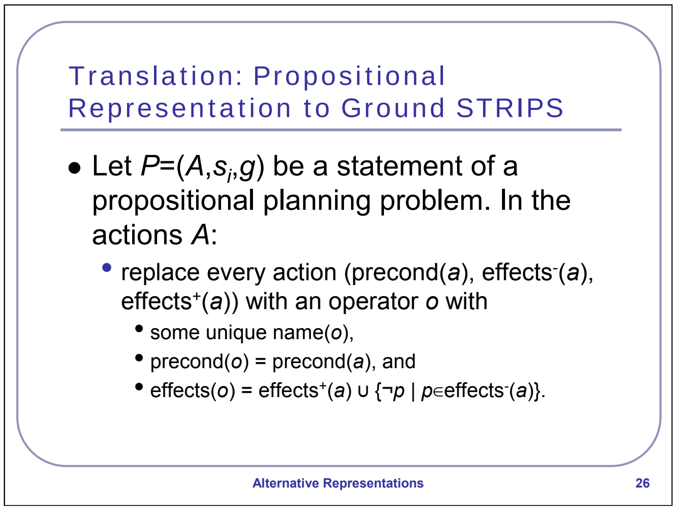
Grounding a STRIPS Planning Problem

•Let P=(O,s_i,g) be the statement of a STRIPS planning problem and C the set of all the constant symbols that are mentioned in s_i. Let ground(O) be the set of all possible instantiations of operators in O with constant symbols from C consistently replacing variables in preconditions and effects.

- •the number of operators will increase exponentially here

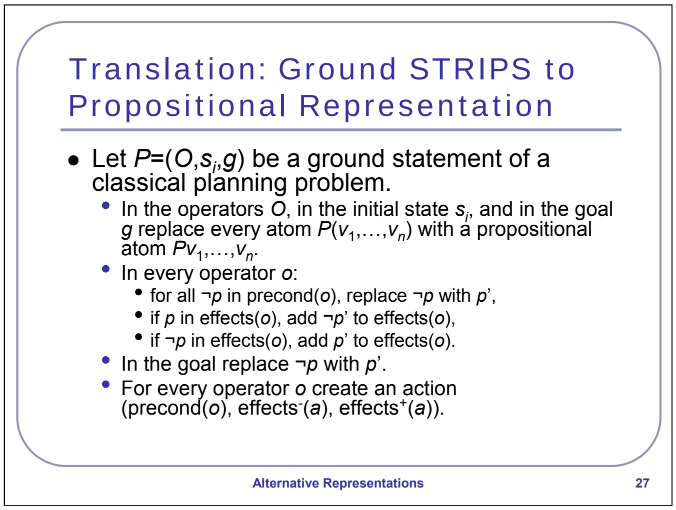
•Then P'=(ground(O),s_i,g) is a statement of a STRIPS planning problem and P' has the same solutions as P.

- •the problems are equivalent (except for exponential increase in size)



Translation: Propositional Representation to Ground STRIPS

- **Let $P=(A,s_i,g)$ be a statement of a propositional planning problem. In the actions A:**
  - **replace every action (precond(a), effects$^-$(a), effects$^+$(a)) with an operator o with**
    - **some unique name(o),**
    - **precond(o) = precond(a), and**
    - **effects(o) = effects$^+$(a) ∪ {¬p | p∈effects$^-$(a)}.**
      - adds negation sign to negative effects
- result is a statement of a ground STRIPS planning problem



Translation: Ground STRIPS to Propositional Representation

•Let P=(O,s_i,g) be a ground statement of a classical planning problem.

•problem: operators may contain negated preconditions

•In the operators O, in the initial state s_i, and in the goal g replace every atom $P(v_1,\ldots,v_n)$ with a propositional atom $Pv_1,\ldots,v_n$.

•idea: introduce new proposition symbols that represent the negations of existing propositions

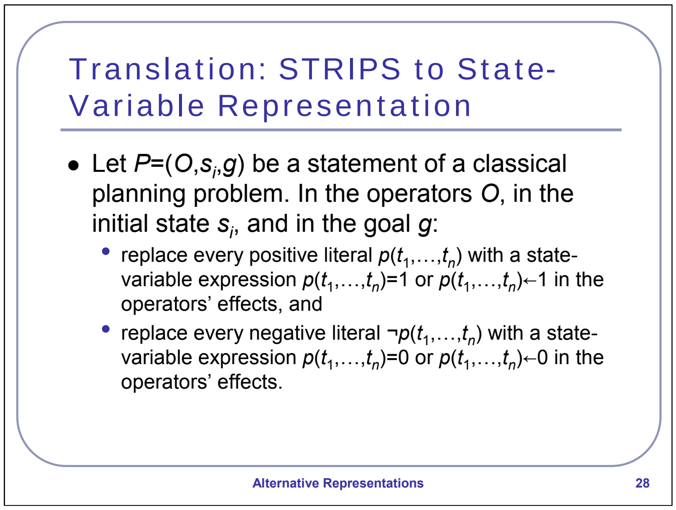
•In every operator *o*:

- **•for all ¬*p*** in **precond(*o*), replace ¬*p* with *p*',**
- **•if *p* in effects(*o*), add ¬*p*' to effects(*o*),**
- **•if ¬*p* in effects(*o*), add *p*' to effects(*o*).**

•In the goal replace ¬*p* with *p*'.

•For every operator *o* create an action (precond(*o*), effects$^-$(*a*), effects$^+$(*a*)).

•result is a statement of a propositional planning problem

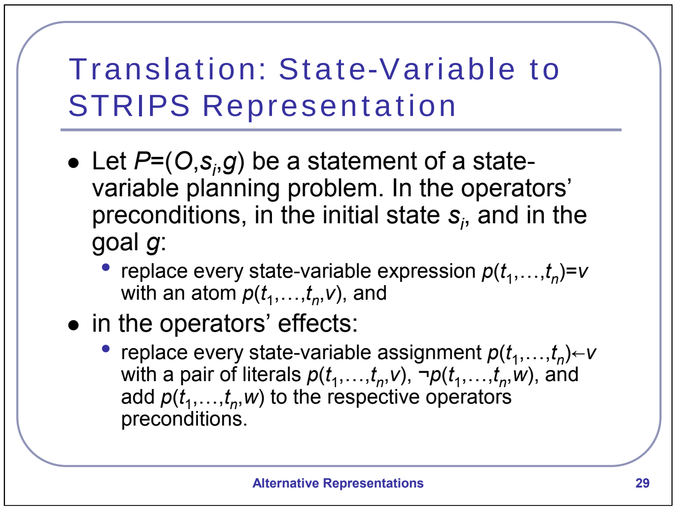


Translation: STRIPS to State-Variable Representation

•Let $P=(O,s_i,g)$ be a statement of a classical planning problem. In the operators O, in the initial state s_i, and in the goal g:

- **•replace every positive literal $p(t_1,\ldots,t_n)$ with a state-variable expression $p(t_1,\ldots,t_n)=1$ or $p(t_1,\ldots,t_n)\leftarrow 1$ in the operators' effects, and**
- **•replace every negative literal $\neg p(t_1,\ldots,t_n)$ with a state-variable expression $p(t_1,\ldots,t_n)=0$ or $p(t_1,\ldots,t_n)\leftarrow 0$ in the operators' effects.**

•result is a statement of a state-variable planning problem



Translation: State-Variable to STRIPS Representation

- **Let P=(O,s_i,g) be a statement of a state-variable planning problem. In the operators' preconditions, in the initial state s_i, and in the goal g:**
  - **replace every state-variable expression $p(t_1,\ldots,t_n)=v$ with an atom $p(t_1,\ldots,t_n,v)$, and**
- **in the operators' effects:**
  - **replace every state-variable assignment $p(t_1,\ldots,t_n)\leftarrow v$ with a pair of literals $p(t_1,\ldots,t_n,v)$, $\neg p(t_1,\ldots,t_n,w)$, and add $p(t_1,\ldots,t_n,w)$ to the respective operators preconditions.**
- result is a statement of a STRIPS planning problem

Overview

World States

•Domains and Operators

•Planning Problems

•Plans and Solutions

•Expressiveness