

# Principles of AI Planning

## 2. Transition systems and planning tasks

Albert-Ludwigs-Universität Freiburg



**UNI  
FREIBURG**

Bernhard Nebel and Robert Mattmüller

October 23rd, 2019



# Transition systems

Transition  
systems

Definition

Blocks world

Planning  
tasks

Summary

## Definition (transition system)

A **transition system** is a 5-tuple  $\mathcal{T} = \langle S, L, T, s_0, S_\star \rangle$  where

- $S$  is a finite set of **states**,
- $L$  is a finite set of (transition) **labels**,
- $T \subseteq S \times L \times S$  is the **transition relation**,
- $s_0 \in S$  is the **initial state**, and
- $S_\star \subseteq S$  is the set of **goal states**.

We say that  $\mathcal{T}$  **has the transition**  $\langle s, l, s' \rangle$  if  $\langle s, l, s' \rangle \in T$ .

We also write this  $s \xrightarrow{l} s'$ , or  $s \rightarrow s'$  when not interested in  $l$ .

**Note:** Transition systems are also called **state spaces**.

Transition systems

Definition

Blocks world

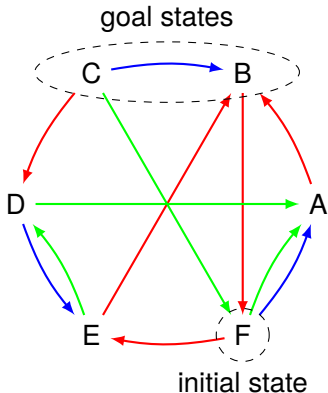
Planning tasks

Summary

# Transition systems: example



Transition systems are often depicted as **directed arc-labeled graphs** with marks to indicate the initial state and goal states.



Transition systems

Definition

Blocks world

Planning tasks

Summary

We use common graph theory terms for transition systems:

- $s'$  **successor** of  $s$  if  $s \rightarrow s'$
- $s$  **predecessor** of  $s'$  if  $s \rightarrow s'$
- $s'$  **reachable** from  $s$  if there exists a sequence of transitions

$$s^0 \xrightarrow{\ell_1} s^1, \dots, s^{n-1} \xrightarrow{\ell_n} s^n \text{ s.t. } s^0 = s \text{ and } s^n = s'$$

- **Note:**  $n = 0$  possible; then  $s = s'$
- $s^0 \xrightarrow{\ell_1} s^1, \dots, s^{n-1} \xrightarrow{\ell_n} s^n$  is called **path** from  $s$  to  $s'$
- $s^0, \dots, s^n$  is also called **path** from  $s$  to  $s'$
- **length** of that path is  $n$
- additional terms: **strongly connected**, **weakly connected**, **strong/weak connected components**, ...

Transition systems

Definition

Blocks world

Planning tasks

Summary

Some additional terminology:

- **$s'$  reachable** (without reference state) means reachable from initial state  $s_0$
- **solution** or **goal path** from  $s$ : path from  $s$  to some  $s' \in S_*$ 
  - if  $s$  is omitted,  $s = s_0$  is implied
- transition system **solvable** if a goal path from  $s_0$  exists

## Definition (deterministic transition system)

A transition system with transitions  $T$  is called **deterministic** if for all states  $s$  and labels  $\ell$ , there is **at most one** state  $s'$  with  $s \xrightarrow{\ell} s'$ .

**Example:** previously shown transition system

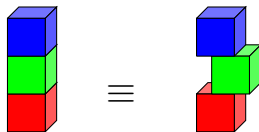
- Throughout the course, we will often use the **blocks world** domain as an example.
- In the blocks world, a number of differently coloured blocks are arranged on our table.
- Our job is to rearrange them according to a given goal.



Location on the table does not matter.

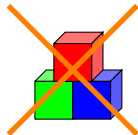


Location on a block does not matter.

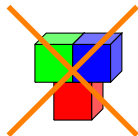


# Blocks world rules (ctd.)

At most one block may be below a block.



At most one block may be on top of a block.



Transition  
systems

Definition

**Blocks world**

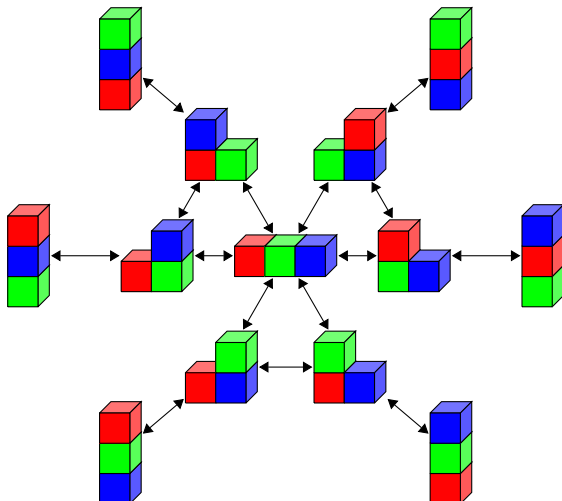
Planning  
tasks

Summary

# Blocks world transition system for three blocks



(Transition labels omitted for clarity.)



Transition systems

Definition

Blocks world

Planning tasks

Summary



blocks	states	blocks	states
1	1	10	58941091
2	3	11	824073141
3	13	12	12470162233
4	73	13	202976401213
5	501	14	3535017524403
6	4051	15	65573803186921
7	37633	16	1290434218669921
8	394353	17	26846616451246353
9	4596553	18	588633468315403843

- Finding a solution is polynomial time in the number of blocks (move everything onto the table and then construct the goal configuration).
- Finding a shortest solution is NP-complete (for a compact description of the problem).



# Planning tasks

Transition  
systems

**Planning  
tasks**

State variables

Logic

Operators

Tasks

Summary

- Classical (i. e., deterministic) planning is in essence the problem of finding solutions in **huge** transition systems.
- The transition systems we are usually interested in are too large to explicitly enumerate all states or transitions.
- Hence, the input to a planning algorithm must be given in a more **concise** form.
- In the rest of chapter, we discuss how to represent planning tasks in a suitable way.

Transition systems

Planning tasks

State variables

Logic

Operators

Tasks

Summary

How to represent huge state sets without enumerating them?

- represent different aspects of the world in terms of different **state variables**

↪ a state is a **valuation of state variables**

- $n$  state variables with  $m$  possible values each induce  $m^n$  different states

↪ **exponentially more compact** than “flat” representations

- **Example:**  $n$  variables suffice for blocks world with  $n$  blocks

Transition  
systems

Planning  
tasks

State variables

Logic

Operators

Tasks

Summary

# Blocks world with finite-domain state variables

Describe blocks world state with three state variables:

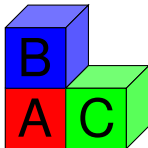
- *location-of-A*: {B, C, table}
- *location-of-B*: {A, C, table}
- *location-of-C*: {A, B, table}

## Example

$s(\textit{location-of-A}) = \textit{table}$

$s(\textit{location-of-B}) = \textit{A}$

$s(\textit{location-of-C}) = \textit{table}$



Not all valuations correspond to intended blocks world states.

**Example:**  $s$  with  $s(\textit{location-of-A}) = \textit{B}$ ,  $s(\textit{location-of-B}) = \textit{A}$ .

Transition  
systems

Planning  
tasks

State variables

Logic

Operators

Tasks

Summary



## Problem:

- How to **succinctly** represent **transitions** and **goal states**?

## Idea: Use **propositional logic**

- **state variables**: propositional variables (0 or 1)
- **goal states**: defined by a propositional formula
- **transitions**: defined by **actions** given by
  - **precondition**: when is the action applicable?
  - **effect**: how does it change the valuation?

**Note:** general finite-domain state variables can be compactly encoded as Boolean variables

Transition  
systems

Planning  
tasks

State variables

Logic

Operators

Tasks

Summary

## Example

$$s(A\text{-on-}B) = 0$$

$$s(A\text{-on-}C) = 0$$

$$s(A\text{-on-table}) = 1$$

$$s(B\text{-on-}A) = 1$$

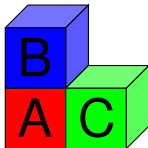
$$s(B\text{-on-}C) = 0$$

$$s(B\text{-on-table}) = 0$$

$$s(C\text{-on-}A) = 0$$

$$s(C\text{-on-}B) = 0$$

$$s(C\text{-on-table}) = 1$$



Transition  
systems

Planning  
tasks

State variables

Logic

Operators

Tasks

Summary

## Definition (propositional formula)

Let  $A$  be a set of **atomic propositions** (here: state variables).

The **propositional formulae** over  $A$  are constructed by finite application of the following rules:

- $\top$  and  $\perp$  are propositional formulae (**truth** and **falsity**).
- For all  $a \in A$ ,  $a$  is a propositional formula (**atom**).
- If  $\varphi$  is a propositional formula, then so is  $\neg\varphi$  (**negation**).
- If  $\varphi$  and  $\psi$  are propositional formulas, then so are  $(\varphi \vee \psi)$  (**disjunction**) and  $(\varphi \wedge \psi)$  (**conjunction**).

**Note:** We often omit the word “propositional”.

Transition  
systems

Planning  
tasks

State variables

Logic

Operators

Tasks

Summary

## Abbreviations:

- $(\varphi \rightarrow \psi)$  is short for  $(\neg\varphi \vee \psi)$  (**implication**)
- $(\varphi \leftrightarrow \psi)$  is short for  $((\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi))$  (**equivalence**)
- parentheses omitted when not necessary
- $(\neg)$  binds more tightly than binary connectives
- $(\wedge)$  binds more tightly than  $(\vee)$  than  $(\rightarrow)$  than  $(\leftrightarrow)$

## Definition (propositional valuation)

A **valuation** of propositions  $A$  is a function  $v : A \rightarrow \{0, 1\}$ .

Define the notation  $v \models \varphi$  ( $v$  **satisfies**  $\varphi$ ;  $v$  is a **model** of  $\varphi$ ;  $\varphi$  is **true** under  $v$ ) for valuations  $v$  and formulae  $\varphi$  by

- $v \models \top$
- $v \not\models \perp$
- $v \models a$  iff  $v(a) = 1$ , for  $a \in A$ .
- $v \models \neg\varphi$  iff  $v \not\models \varphi$
- $v \models \varphi \vee \psi$  iff  $v \models \varphi$  or  $v \models \psi$
- $v \models \varphi \wedge \psi$  iff  $v \models \varphi$  and  $v \models \psi$

Transition  
systems

Planning  
tasks

State variables

Logic

Operators

Tasks

Summary

- A propositional formula  $\varphi$  is **satisfiable** if there is at least one valuation  $v$  so that  $v \models \varphi$ .
- Otherwise it is **unsatisfiable**.
- A propositional formula  $\varphi$  is **valid** or a **tautology** if  $v \models \varphi$  for all valuations  $v$ .
- A propositional formula  $\psi$  is a **logical consequence** of a propositional formula  $\varphi$ , written  $\varphi \models \psi$ , if  $v \models \psi$  for all valuations  $v$  with  $v \models \varphi$ .
- Two propositional formulae  $\varphi$  and  $\psi$  are **logically equivalent**, written  $\varphi \equiv \psi$ , if  $\varphi \models \psi$  and  $\psi \models \varphi$ .

**Question:** How to phrase these in terms of **models**?

Transition  
systems

Planning  
tasks

State variables

Logic

Operators

Tasks

Summary



- A propositional formula that is a proposition  $a$  or a negated proposition  $\neg a$  for some  $a \in A$  is a **literal**.
- A formula that is a disjunction of literals is a **clause**. This includes **unit clauses** / consisting of a single literal, and the **empty clause**  $\perp$  consisting of zero literals.

**Normal forms:** NNF, CNF, DNF

Transition systems

Planning tasks

State variables

Logic

Operators

Tasks

Summary

Transitions for state sets described by propositions  $A$  can be concisely represented as **operators** or **actions**  $\langle \chi, e \rangle$  where

- the **precondition**  $\chi$  is a propositional formula over  $A$  describing the set of states in which the transition can be taken (states in which a transition starts), and
- the **effect**  $e$  describes how the resulting successor states are obtained from the state where the transitions is taken (where the transition goes).

Transition systems

Planning tasks

State variables

Logic

Operators

Tasks

Summary



## Blocks world operators

To model blocks world operators conveniently, we use auxiliary state variables *A-clear*, *B-clear*, and *C-clear* to denote that there is nothing on top of a given block.

Then blocks world operators can be modeled as:

- $\langle A\text{-clear} \wedge A\text{-on-}T \wedge B\text{-clear}, A\text{-on-}B \wedge \neg A\text{-on-}T \wedge \neg B\text{-clear} \rangle$
- $\langle A\text{-clear} \wedge A\text{-on-}T \wedge C\text{-clear}, A\text{-on-}C \wedge \neg A\text{-on-}T \wedge \neg C\text{-clear} \rangle$
- $\langle A\text{-clear} \wedge A\text{-on-}B, A\text{-on-}T \wedge \neg A\text{-on-}B \wedge B\text{-clear} \rangle$
- $\langle A\text{-clear} \wedge A\text{-on-}C, A\text{-on-}T \wedge \neg A\text{-on-}C \wedge C\text{-clear} \rangle$
- $\langle A\text{-clear} \wedge A\text{-on-}B \wedge C\text{-clear}, A\text{-on-}C \wedge \neg A\text{-on-}B \wedge B\text{-clear} \wedge \neg C\text{-clear} \rangle$
- $\langle A\text{-clear} \wedge A\text{-on-}C \wedge B\text{-clear}, A\text{-on-}B \wedge \neg A\text{-on-}C \wedge C\text{-clear} \wedge \neg B\text{-clear} \rangle$
- ...

## Definition (effects)

(Deterministic) **effects** are recursively defined as follows:

- If  $a \in A$  is a state variable, then  $a$  and  $\neg a$  are effects (**atomic effect**).
- If  $e_1, \dots, e_n$  are effects, then  $e_1 \wedge \dots \wedge e_n$  is an effect (**conjunctive effect**).  
The special case with  $n = 0$  is the empty effect  $\top$ .
- If  $\chi$  is a propositional formula and  $e$  is an effect, then  $\chi \triangleright e$  is an effect (**conditional effect**).

Atomic effects  $a$  and  $\neg a$  are best understood as assignments  $a := 1$  and  $a := 0$ , respectively.

Transition  
systems

Planning  
tasks

State variables

Logic

Operators

Tasks

Summary

$\chi \triangleright e$  means that change  $e$  takes place if  $\chi$  is true in the current state.

## Example

Increment 4-bit number  $b_3b_2b_1b_0$  represented as four state variables  $b_0, \dots, b_3$ :

$$\begin{aligned} & (\neg b_0 \triangleright b_0) \wedge \\ & ((\neg b_1 \wedge b_0) \triangleright (b_1 \wedge \neg b_0)) \wedge \\ & ((\neg b_2 \wedge b_1 \wedge b_0) \triangleright (b_2 \wedge \neg b_1 \wedge \neg b_0)) \wedge \\ & ((\neg b_3 \wedge b_2 \wedge b_1 \wedge b_0) \triangleright (b_3 \wedge \neg b_2 \wedge \neg b_1 \wedge \neg b_0)) \end{aligned}$$

Transition  
systems

Planning  
tasks

State variables

Logic

Operators

Tasks

Summary

## Definition (changes caused by an operator)

For each effect  $e$  and state  $s$ , we define the **change set** of  $e$  in  $s$ , written  $[e]_s$ , as the following set of literals:

- $[a]_s = \{a\}$  and  $[\neg a]_s = \{\neg a\}$  for atomic effects  $a$ ,  $\neg a$
- $[e_1 \wedge \dots \wedge e_n]_s = [e_1]_s \cup \dots \cup [e_n]_s$
- $[\chi \triangleright e]_s = [e]_s$  if  $s \models \chi$  and  $[\chi \triangleright e]_s = \emptyset$  otherwise

## Definition (applicable operators)

Operator  $\langle \chi, e \rangle$  is **applicable in a state  $s$**  iff  $s \models \chi$  and  $[e]_s$  is consistent (i. e., does not contain two complementary literals).

Transition  
systems

Planning  
tasks

State variables

Logic

Operators

Tasks

Summary

## Definition (successor state)

The **successor state**  $app_o(s)$  of  $s$  with respect to operator  $o = \langle \chi, e \rangle$  is the state  $s'$  with  $s' \models [e]_s$  and  $s'(v) = s(v)$  for all state variables  $v$  not mentioned in  $[e]_s$ .

This is defined only if  $o$  is applicable in  $s$ .

## Example

Consider the operator  $\langle a, \neg a \wedge (\neg c \triangleright \neg b) \rangle$  and the state  $s = \{a \mapsto 1, b \mapsto 1, c \mapsto 1, d \mapsto 1\}$ .

The operator is applicable because  $s \models a$  and  $[\neg a \wedge (\neg c \triangleright \neg b)]_s = \{\neg a\}$  is consistent.

Applying the operator results in the successor state  $app_{\langle a, \neg a \wedge (\neg c \triangleright \neg b) \rangle}(s) = \{a \mapsto 0, b \mapsto 1, c \mapsto 1, d \mapsto 1\}$ .

Transition systems

Planning tasks

State variables

Logic

Operators

Tasks

Summary

## Definition (deterministic planning task)

A **deterministic planning task** is a 4-tuple  $\Pi = \langle A, I, O, \gamma \rangle$  where

- $A$  is a finite set of **state variables** (propositions),
- $I$  is a valuation over  $A$  called the **initial state**,
- $O$  is a finite set of **operators** over  $A$ , and
- $\gamma$  is a formula over  $A$  called the **goal**.

## Note:

- When we talk about deterministic planning tasks, we usually omit the word “deterministic”.
- When we will talk about nondeterministic planning tasks later, we will explicitly qualify them as “nondeterministic”.

Transition  
systems

Planning  
tasks

State variables

Logic

Operators

Tasks

Summary



## Definition (induced transition system of a planning task)

Every planning task  $\Pi = \langle A, I, O, \gamma \rangle$  induces a corresponding deterministic transition system  $\mathcal{T}(\Pi) = \langle S, L, T, s_0, S_\star \rangle$ :

- $S$  is the set of all valuations of  $A$ ,
- $L$  is the set of operators  $O$ ,
- $T = \{ \langle s, o, s' \rangle \mid s \in S, o \text{ applicable in } s, s' = \text{app}_o(s) \}$ ,
- $s_0 = I$ , and
- $S_\star = \{ s \in S \mid s \models \gamma \}$

Transition systems

Planning tasks

State variables

Logic

Operators

Tasks

Summary

- Terminology for transitions systems is also applied to the planning tasks that induce them.
- For example, when we speak of the **states of  $\Pi$** , we mean the states of  $\mathcal{T}(\Pi)$ .
- A sequence of operators that forms a goal path of  $\mathcal{T}(\Pi)$  is called a **plan** of  $\Pi$ .

Transition  
systems

Planning  
tasks

State variables

Logic

Operators

Tasks

Summary



By **planning**, we mean the following two algorithmic problems:

## Definition (satisficing planning)

**Given:** a planning task  $\Pi$

**Output:** a plan for  $\Pi$ , or **unsolvable** if no plan for  $\Pi$  exists

## Definition (optimal planning)

**Given:** a planning task  $\Pi$

**Output:** a plan for  $\Pi$  with minimal length among all plans for  $\Pi$ , or **unsolvable** if no plan for  $\Pi$  exists

Transition  
systems

Planning  
tasks

State variables

Logic

Operators

Tasks

Summary



- **Transition systems** are (typically huge) directed graphs that encode how the state of the world can change.
- **Planning tasks** are compact representations for transition systems, suitable as input for planning algorithms.
- Planning tasks are based on concepts from **propositional logic**, enhanced to model state change.
- **States** of planning tasks are propositional valuations.
- **Operators** of planning tasks describe **when** (precondition) and **how** (effect) to change the current state of the world.
- In **satisficing planning**, we must find a solution to planning tasks (or show that no solution exists).
- In **optimal planning**, we additionally guarantee that generated solutions are of the shortest possible length.

Transition  
systems

Planning  
tasks

Summary