

Planning and Optimization

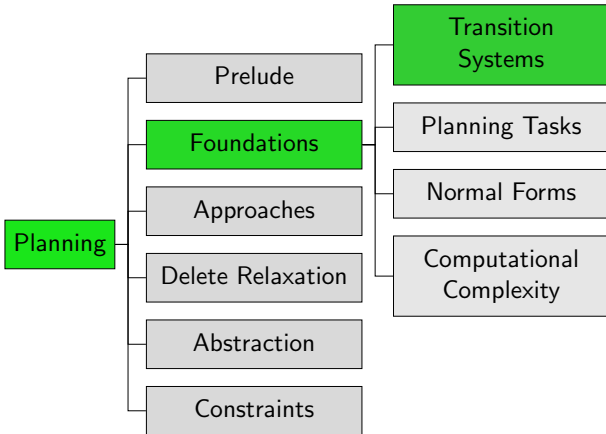
B1. Transition Systems and Propositional Logic

Malte Helmert and Gabriele Röger

Universität Basel

September 21, 2026

Content of the Course



Next Steps

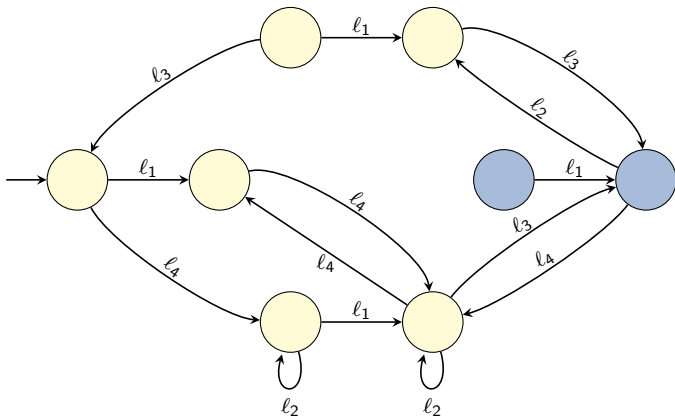
Our next steps are to formally define our problem:

- introduce a mathematical model for planning tasks:
transition systems
⇒ Chapter B1
- introduce **compact representations** for planning tasks
suitable as input for planning algorithms
⇒ Chapter B2

Transition Systems

Transition System Example

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



$$c(\ell_1) = 1, c(\ell_2) = 1, c(\ell_3) = 5, c(\ell_4) = 0$$

Transition Systems

Definition (Transition System)

A **transition system** is a 6-tuple $\mathcal{T} = \langle S, L, c, T, s_0, S_\star \rangle$ where

- S is a finite set of **states**,
- L is a finite set of (transition) **labels**,
- $c : L \rightarrow \mathbb{R}_0^+$ is a **label cost** function,
- $T \subseteq S \times L \times S$ 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** $\langle s, \ell, s' \rangle$ if $\langle s, \ell, s' \rangle \in T$.

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

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

Deterministic Transition Systems

Definition (Deterministic Transition System)

A transition system is called **deterministic** if for all states s and all labels ℓ , there is **at most one** state s' with $s \xrightarrow{\ell} s'$.

Example: previously shown transition system

Transition System Terminology (1)

We use common terminology from graph theory:

- s' **successor** of s if $s \rightarrow s'$
- s **predecessor** of s' if $s \rightarrow s'$

Transition System Terminology (2)

We use common terminology from graph theory:

- 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 $s = s'$
- $s^0, \dots, s^n$ is called **(state) path** from s to s'
- $\ell_1, \dots, \ell_n$ is called **(label) path** from s to s'
- $s^0 \xrightarrow{\ell_1} s^1, \dots, s^{n-1} \xrightarrow{\ell_n} s^n$ is called **trace** from s to s'
- **length** of path/trace is n
- **cost** of label path/trace is $\sum_{i=1}^n c(\ell_i)$

Transition System Terminology (3)

We use common terminology from graph theory:

- 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

Example: Blocks World

Running Example: Blocks World

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

Blocks World Rules (1)

Location on the table does not matter.

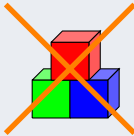


Location on a block does not matter.

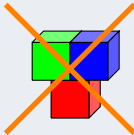


Blocks World Rules (2)

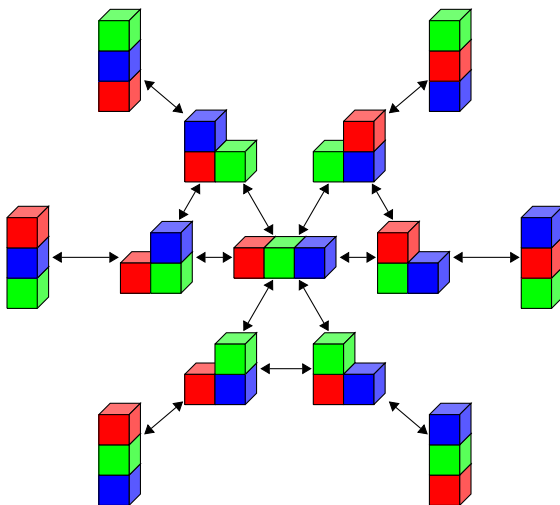
At most one block may be below a block.



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



Blocks World Transition System for Three Blocks



Labels omitted for clarity. All label costs are 1. Initial/goal states not marked.

Blocks World Computational Properties

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 solutions is possible in linear time in the number of blocks: move everything onto the table, then construct the goal configuration.
- Finding a shortest solution is NP-complete given a compact description of the problem.

The Need for Compact Descriptions

- We see from the blocks world example that transition systems are often **far too large** to be directly used as **inputs** to planning algorithms.
- We therefore need **compact descriptions** of transition systems.
- For this purpose, we will use **propositional logic**, which allows expressing information about 2^n states as logical formulas over n **state variables**.

Reminder: Propositional Logic

More on Propositional Logic

Need to Catch Up?

- This section is a **reminder**. We assume you are already well familiar with propositional logic.
- If this is not the case, we recommend Chapters D1–D4 of the **Discrete Structures** (formerly **Discrete Mathematics in Computer Science**) course:
<https://dmi.unibas.ch/en/studies/computer-science/course-offer-fall-semester-25/lecture-discrete-mathematics-in-computer-science/>
 - Videos for these chapters are available on request.

Syntax of Propositional Logic

Definition (Logical Formula)

Let A be a set of **atomic propositions**.

The **logical formulas** over A are constructed by finite application of the following rules:

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

Syntactical Conventions for Propositional Logic

Abbreviations:

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

Semantics of Propositional Logic

Definition (Interpretation, Model)

An **interpretation** of propositions A is a function $I : A \rightarrow \{\mathbf{T}, \mathbf{F}\}$.

Define the notation $I \models \varphi$ (I **satisfies** φ ; I is a **model** of φ ; φ is **true** under I) for interpretations I and formulas φ by

- $I \models \top$
- $I \not\models \perp$
- $I \models a$ iff $I(a) = \mathbf{T}$ (for all $a \in A$)
- $I \models \neg\varphi$ iff $I \not\models \varphi$
- $I \models (\varphi \vee \psi)$ iff ($I \models \varphi$ or $I \models \psi$)
- $I \models (\varphi \wedge \psi)$ iff ($I \models \varphi$ and $I \models \psi$)

Note: Interpretations are also called **valuations**
or **truth assignments**.

Propositional Logic Terminology (1)

- A logical formula φ is **satisfiable** if there is at least one interpretation I such that $I \models \varphi$.
- Otherwise it is **unsatisfiable**.
- A logical formula φ is **valid** or a **tautology** if $I \models \varphi$ for all interpretations I .
- A logical formula ψ is a **logical consequence** of a logical formula φ , written $\varphi \models \psi$, if $I \models \psi$ for all interpretations I with $I \models \varphi$.
- Two logical formulas φ and ψ are **logically equivalent**, written $\varphi \equiv \psi$, if $\varphi \models \psi$ and $\psi \models \varphi$.

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

Propositional Logic Terminology (2)

- A logical formula that is a proposition a or a negated proposition $\neg a$ for some atomic proposition $a \in A$ is a **literal**.
- A formula that is a disjunction of literals is a **clause**.
This includes **unit clauses** ℓ consisting of a single literal and the **empty clause** $\perp$ consisting of zero literals.
- A formula that is a conjunction of literals is a **monomial**.
This includes **unit monomials** ℓ consisting of a single literal and the **empty monomial** $\top$ consisting of zero literals.

Normal forms:

- negation normal form (NNF)
- conjunctive normal form (CNF)
- disjunctive normal form (DNF)

Summary

Summary

- **Transition systems** are (typically huge) directed graphs that encode how the state of the world can change.
- **Propositional logic** allows us to compactly describe complex information about large sets of interpretations as **logical formulas**.